NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPLEMENTATION OF DATA FLOW QUERY LANGUAGE ON A HANDHELD DEVICE

by

Mark A. Evangelista

March 2005

Thesis Advisor:	Thomas W. Otani
Second Reader:	Arijit Das

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Implementation of Data Flow Query Language on a Handheld Device		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Evangelista, Mark A.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Handheld devices have evolved significantly from mere simple organizers to more powerful handheld computers that are capable of network connectivity, giving it the ability to send e-mail, browse the World Wide Web, and query remote databases. However, handheld devices, because of its design philosophy, are limited in terms of size, memory, and processing power compared to desktop computers.

This thesis investigates the use of Data Flow Query Language (DFQL) in querying local and remote databases from a handheld device. Creating Standard Query Language (SQL) queries can be a complex undertaking; and trying to create one on a handheld device with a small screen only adds to its complexity. However, by using DFQL, the user can submit queries with an easy to use graphical user interface.

Although handheld devices are currently more powerful than earlier PCs, they still require applications with a small footprint, which is a limiting factor for software developed. This thesis will also investigate the best division of labor between handheld device and remote servers.

14. SUBJECT TERMS Structure Query, SQL, Data Flow Query Language, DFQL, .NET Compact Framework, Object Oriented Programming, Wireless Network Communication, Handheld application			**15. NUMBER OF PAGES** 191
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

IMPLEMENTATION OF DATA FLOW QUERY LANGUAGE ON A HANDHELD DEVICE

Mark A. Evangelista
Sergeant, United States Army
B.S., De La Salle University, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2005

Author: Mark A. Evangelista

Approved by: Thomas W. Otani
Thesis Advisor

Arijit Das
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Handheld devices have evolved significantly from mere simple organizers to more powerful handheld computers that are capable of network connectivity, giving it the ability to send e-mail, browse the World Wide Web, and query remote databases. However, handheld devices, because of its design philosophy, are limited in terms of size, memory, and processing power compared to desktop computers.

This thesis investigates the use of Data Flow Query Language (DFQL) in querying local and remote databases from a handheld device. Creating Standard Query Language (SQL) queries can be a complex undertaking; and trying to create one on a handheld device with a small screen only adds to its complexity. However, by using DFQL, the user can submit queries with an easy to use graphical user interface.

Although handheld devices are currently more powerful than earlier PCs, they still require applications with a small footprint, which is a limiting factor for software developed. This thesis will also investigate the best division of labor between handheld device and remote servers.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank the US Army for giving me the opportunity to be all I can be. Also, I would like to thank the US Navy, most especially, the Naval Postgraduate School, to allow enlisted personnel, like myself, the opportunity to prove themselves capable of attaining a Master's degree.

I am especially indebted to SFC Ferris and SPC Stevens, without them I would have not known I could take graduate courses at NPS.

I sincerely thank Mrs. Kastner and MAJ Phillips for their support. They both allowed me to pursue a degree at NPS while working at MPD and TRAC-Monterey. Thanks also to LTC Cioppa and Mr. Jackson, who constantly made sure I was progressing with my thesis.

I also greatly appreciate all the time Professor Otani and Mr. Das had spent with me sharing their insight and knowledge regarding many design and implementation decisions.

Finally, but not least, I would like to thank my loving wife Miki for her enduring patience, support, and sacrifice; and my two sons, Marcus and Mateo, for being my inspiration.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

Handheld devices have evolved significantly from mere simple organizers to more powerful handheld computers that are capable of network connectivity, giving it the ability to send e-mail, browse the World Wide Web, and query remote databases. However, handheld devices, because of its design philosophy, are limited in terms of size, memory, and processing power compared to desktop computers.

This thesis investigates the use of Data Flow Query Language (DFQL) in querying local and remote databases from a handheld device. Creating Standard Query Language (SQL) queries can be a complex undertaking; and trying to create one on a handheld device with a small screen only adds to its complexity. However, by using DFQL, the user can submit queries with an easy to use graphical user interface.

Although handheld devices are currently more powerful than earlier PCs, they still require applications with a small footprint, which is a limiting factor for software developed. This thesis will also investigate the best division of labor between handheld device and remote servers.

B. HANDHELD DEVICES

Handheld devices are small wireless electronic devices that are capable of sending, receiving, storing, and displaying data. Cell phones, Personal Digital Assistants (PDAs), and Blackberrys™ are some examples of handheld devices. These devices are miniaturized versions of desktop computers.

Applications that are developed for desktop computers are finding their ways into handheld devices. Cell phones are now capable of surfing the web. Blackberry™ devices are capable of sending, receiving corporate e-mails, and viewing attached documents. PDAs have office applications that are similar to

desktop computers that are capable of reading documents, spreadsheets, and databases.

However, all handheld devices continue to have cramped and inconvenient inputting keys. Cell phones have tedious multifunction keys and navigational buttons, PDAs have touch pens and Software Input Panels, and Blackberrys™ have scrolling wheels and built-in keyboard, but none can compete with the ease and speed of using a standard keyboard and mouse.

C. DATABASES

Database applications are excellent candidates for mobile applications. With a few inputs, a user is capable of retrieving huge amounts of data. With a few taps, returned data can come as either a one line short text or large amounts displayed in a table. The ideal mobile application would be for users to access data with the least amount of user input.

However, underneath any relational database application is the SQL language. SQL is a text based query language that was invented by IBM™, and is the most widely used query language for relational databases [Ref.1:p.324]

In addition to being text based, SQL statements can become complex. Part of the problem is its declarative nature. All the conditions that a query result must meet are specified in a single statement, rather than in a sequence of statements. This problem is made worse when complex queries involve universal quantification due to its negative logic implementation in SQL. Universal quantification is supported only indirectly in SQL. Not making it any easier, the logical meaning of universal quantification is not completely intuitive. Especially to persons who are not experienced in the use of predicate logic, this idea of indirect support adds only to its complexity[Ref. 2].

SQL has other problems. For example, SQL is a mixture of both relational algebra and calculus with some other ideas thrown in as well. However, due to this mixture, SQL is a strict implementation of neither relational algebra nor calculus. Additionally, SQL lacks orthogonality. This means that there is a

relatively small set of primitives that can be combined in a relatively small number of ways to build the control and data structure of the language. It increases the number of special rules that must be memorized by the user, decreases the readability and writability of the language, and in general the usability of the language[Ref. 2].

It is because of the difficulties of SQL that DFQL was created. DFQL was first introduced by Gard J. Clark, C. Thomas Wu, Naval Postgraduate School, Department of Computer Science in September 1991. Experiments were made to determine whether DFQL made a significant difference in creating queries. Clark concluded that DFQL produced a significantly higher percentage of correct answers on the more difficult queries[Ref. 2].

DFQL is a great candidate for a handheld application. It is a database application that will require minimal user input and yet capable of accessing large amounts of information.

THIS PAGE INTENTIONALLY LEFT BLANK

II. DATA FLOW QUERY LANGUAGE

A. DESCRIPTION

DFQL is a visual relational algebra to be used for the manipulation of relational databases. It provides the user with a simple yet powerful tool to implement database queries at all levels of complexity [Ref. 2].

Visually, DFQL is a compilation of DFQL operator icons connected with each other. Each operator icon represents a function with parameters and returns, called input and output nodes respectively. A user constructs a DFQL query by choosing which operators to use, and then connects them using the input and output nodes. As a constraint, there is a one to many relationship between output and input nodes. For example, an output node can connect to many inputs while an input node can only connect to one output node (see figure 19).

Input nodes can either receive a *Relation* or *Criteria*. *Relations* can either be a based relation (i.e. Table operator) or a derived relation (i.e., Basic or Advanced operators). *Criterias* are user inputs that are either attributes or conditions.

B. OPERATORS

All DFQL operators appear alike. Operators have a rectangular shape with several nodes. All operators will have a body, output node, move node, delete node, and centered descriptive text. However, some operators will either have no input nodes or 2 to 4 input nodes. Operators representing a Table or Criteria will have no input nodes. While operators with input nodes, ranging from 2 to 4, are considered DFQL operators.

Nodes are color coded. This helps the user distinguish what the nodes represent. The output node is dark blue. Input nodes can either be green or red. Green input node means it will accept a *Relation.* Red on the other hand means

it will accept a *Criteria* operator. Move nodes are color blue, and delete nodes are color black.

Discriptive text found in the center of an operator either displays the type of operator or captured user input data. For example, Table and Criteria operators will display type but will display user created data when available. Other operators will just display operator type (e.g., Select, Project, Intersect).

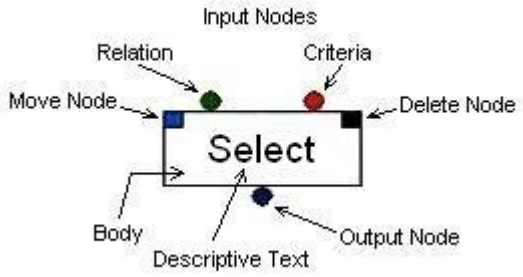


Figure 1. DFQL Operator Map

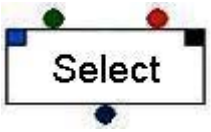
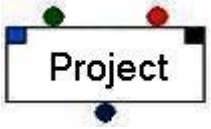
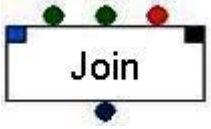
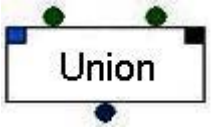
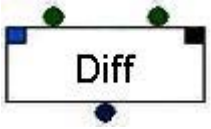
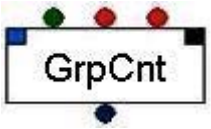
Different from other implementations of DFQL is the absence of a display operator. Display operators had no output nodes and were used as terminators to display results. For this implementation, rather than a display operator, a user can just tap on an output node and the *Results* page will automatically appear to display the result.

Operators with input nodes are DFQL operators, There are two types of DFQL operators. They are the *Basic* and *Advanced* operators.

1. Basic

Basic operators are derived from the requirements for relational completeness. A query language that is complete has the expressive power of first-order predicate calculus. The requirements for relational completeness are the following: *selection*, *projection*, *union*, *difference*, and *Cartesian product (join)*. One *Basic* operator added but not required for completeness is *group count*. Group count is a form of grouping or aggregation.

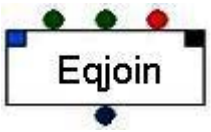
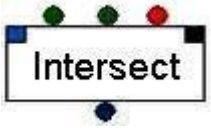
Table 1. DFQL Basic Operators

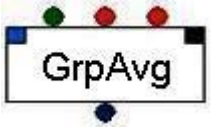
DFQL Operator	SQL Equivalent
 SELECT	SELECT DISTINCT * FROM green input WHERE red input
 PROJECT	SELECT DISTINCT red input FROM green input
 JOIN	SELECT DISTINCT * FROM green input 1, green input 2 WHERE red input
 UNION	SELECT DISTINCT * FROM green input 1 UNION SELECT DISTINCT * FROM green input 2
 DIFFERENCE	SELECT DISTINCT * FROM green input 1 MINUS SELECT DISTINCT * FROM green input 2
 GROUP COUNT	SELECT DISTINCT red input 1 COUNT (*) red input 2 FROM green input GROUP BY red input 1

2. Advanced

Other built-in operators where also introduced by Gard J. Clark. Even though the basic operators are already relationally complete, creating other built-in operators helps to reduce the overhead required when just using *Basic* operators to interpret the query. For example, one operator introduced is the *intersect* operator. Relational intersection can be defined in terms of *union* and *diff* $R_1 \cap R_2 \equiv (R_1 \cup R_2) - ((R_1 - R_2) \cup (R_2 - R_1))$. Rather than create the query in terms of *union* and *diff*, we can take advantage of Database Management System (DBMS) built in functions [Ref. 2]. Many DBMS's provide a specific *intersect* operator and it would be wise to create one for DFQL.

Table 2. Advanced DFQL Operators

DFQL Operator	SQL Equivalent
 EQJOIN	SELECT DISTINCT * FROM green input 1, green input 2 WHERE red input
 GROUP ALL SATISFY	SELECT DISTINCT red input 1 FROM green input WHERE red input 2 GROUP BY red input 1
 GROUP N SATISFY	SELECT DISTINCT red input 1 FROM green input WHERE red input 2 GROUP BY red input 1 HAVING COUNT(*) red input 3
 INTERSECT	SELECT DISTINCT red input FROM green input 1 WHERE red input IN (SELECT red input FROM green input 2)

 GROUP MINIMUM	SELECT DISTINCT red input 1, MIN (red input 2) as MIN_red input 2 FROM green input GROUP BY red input 1
 GROUP MAXIMUM	SELECT DISTINCT red input 1, MAX (red input 2) as MAX_red input 2 FROM green input GROUP BY red input 1
 GROUP AVERAGE	SELECT DISTINCT red input 1, AVG (red input 2) as AVG_RED input 2 FROM green input GROUP BY red input 1

C. QUERY CONSTRUCTION

When creating a DFQL query, the user must first connect to a database. In order to do this, the user taps on the *connect* menu and chooses either a *local* or *remote* database. Once the user connects to at least one database, the user chooses from a list of operators and taps on the screen to choose the location of the operator to be created and displayed. The user is required to make sure that all input nodes are terminated by either a *Relation*, or *Criteria* operator. In order to connect an input node to an output node, the user taps on the input node and then taps on the body of another operator. This creates a line connecting input and output node together. When an operator needs to be relocated, the user taps and holds on the move node. The user then drags and releases the operator to its new location. When an operator needs to be deleted, the user just taps on the delete node and the operator disappears. When all input nodes are terminated and all *Relations* and *Criterias* have data in them, the user taps on an output node to display the result.

1. Table Operator

The user chooses the *Table* operator from the *Operators* submenu and then taps on the screen for the *Table* operator to appear. In order to activate the *Table* operator and connect to a table in a database, the user taps the body of the operator which makes a *Table* form appear. The *Table* form has a tree view of the different databases that the application is connected to.

Figure 2. Table Operator

Figure 3. Table Form

2. Criteria Operator

The user chooses the *Criteria* operator from the *Operators* submenu and then taps on the screen for the *Criteria* operator to appear. In order to activate the *Criteria* operator, the user taps the body of the operator which makes a *Criteria* form appear. The *Criteria* form has a tree view to help the user create criteria for the operator. When the user clicks on a node the text automatically fills the text box with the node tapped. This helps minimize text entry.

Figure 4. Criteria Operator

Figure 5. Criteria Form

3. Select DFQL Operator

The user chooses the *Select* operator from the *Basic* operators submenu and then taps on the screen for the *Select* operator to appear. *Select* operators have two input nodes. One accepts a *Relation* while the other accepts a *Criteria*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

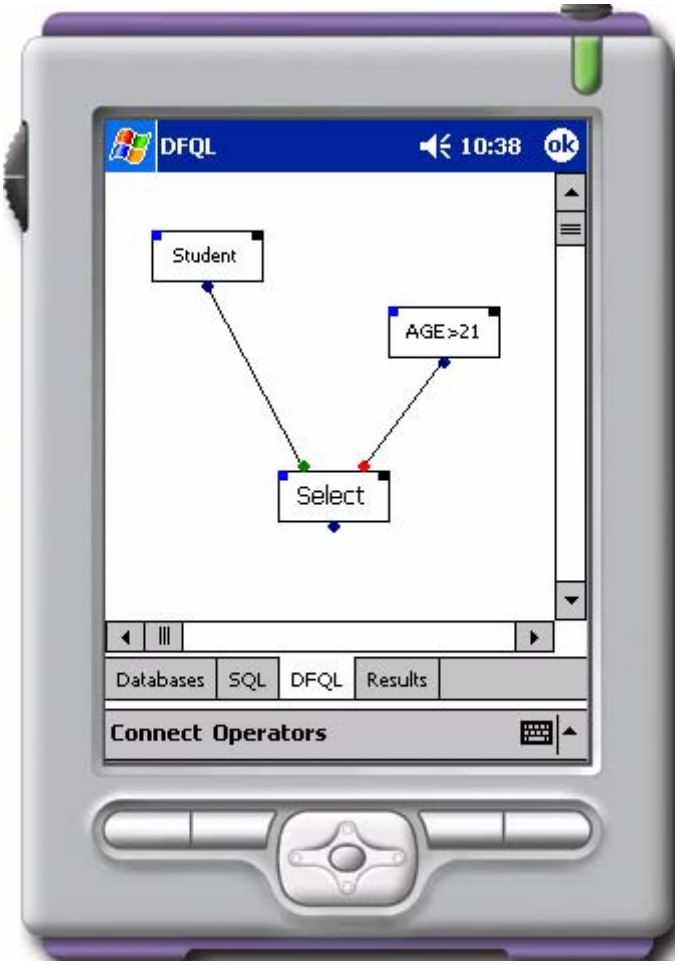


Figure 6. Select DFQL Operator query

The SQL translation of the DFQL query above is: *SELECT * FROM Student WHERE age>21.*

4. Project DFQL Operator

The user chooses the *Project* operator from the *Basic* Operators submenu and then taps on the screen for the *Project* operator to appear. *Project* Operators have two input nodes. One accepts a *Relation* while the other accepts a *Criteria*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

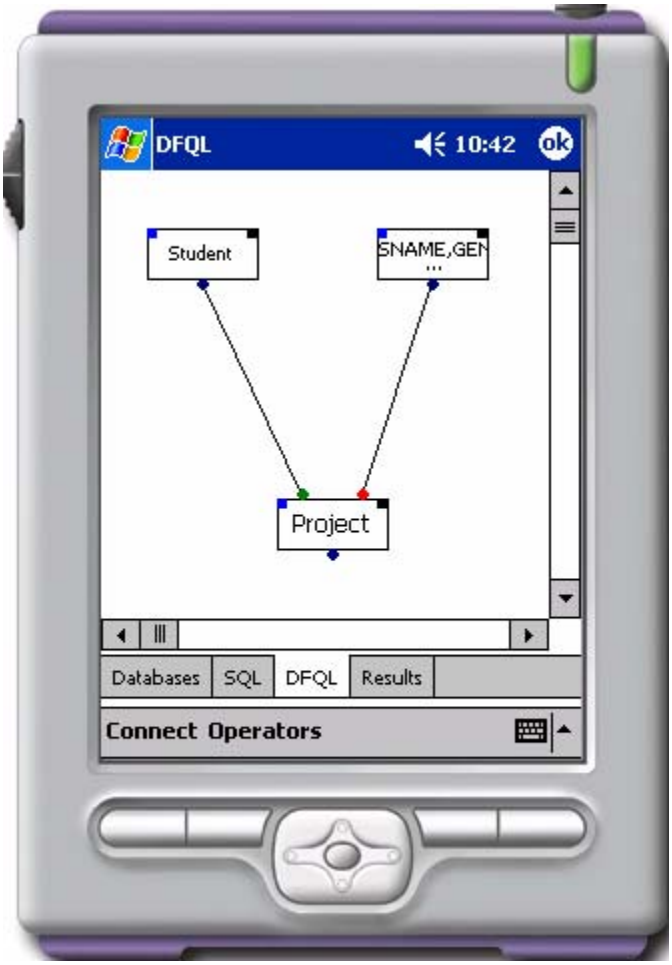


Figure 7. Project DFQL Operator

The SQL translation of the DFQL query above is: *SELECT sname, gender FROM student*.

5. Join DFQL Operator

The user chooses the *Join* operator from the *Basic* operators submenu and then taps on the screen for the *Join* operator to appear. *Join* operators have three input nodes. Two input nodes accepts a *Relation*. The other input accepts a *Criteria*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

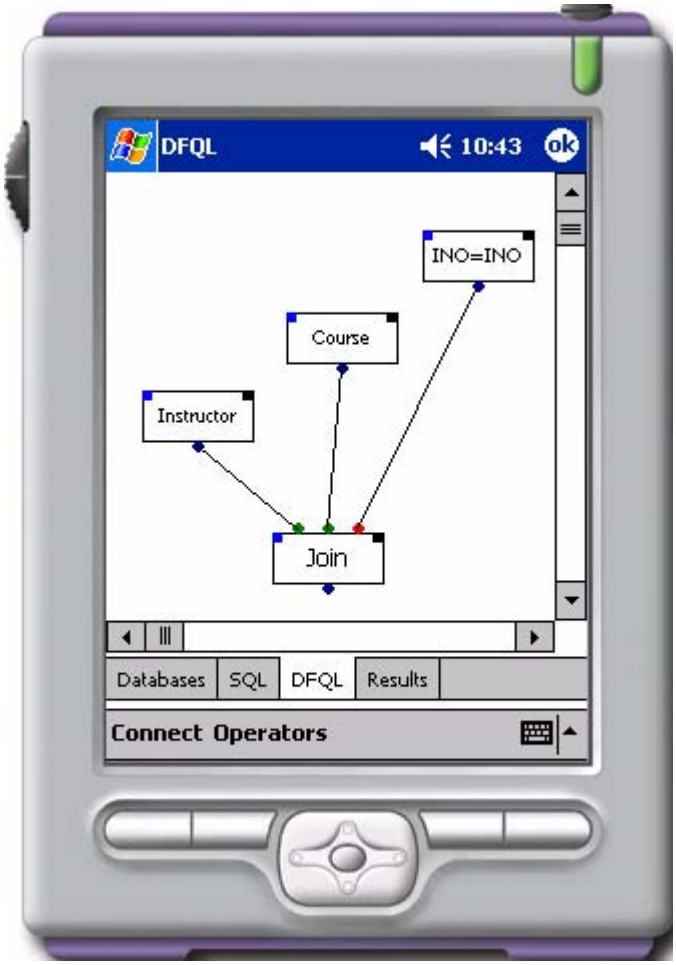


Figure 8. Join DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT * FROM Instructor, Course where Instructor.INO=Course.INO*.

6. Union DFQL Operator

The user chooses the *Union* operator from the *Basic* operators submenu and then taps on the screen for the *Union* operator to appear. *Union* operators have two input nodes. Both input nodes accepts a *Relation*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

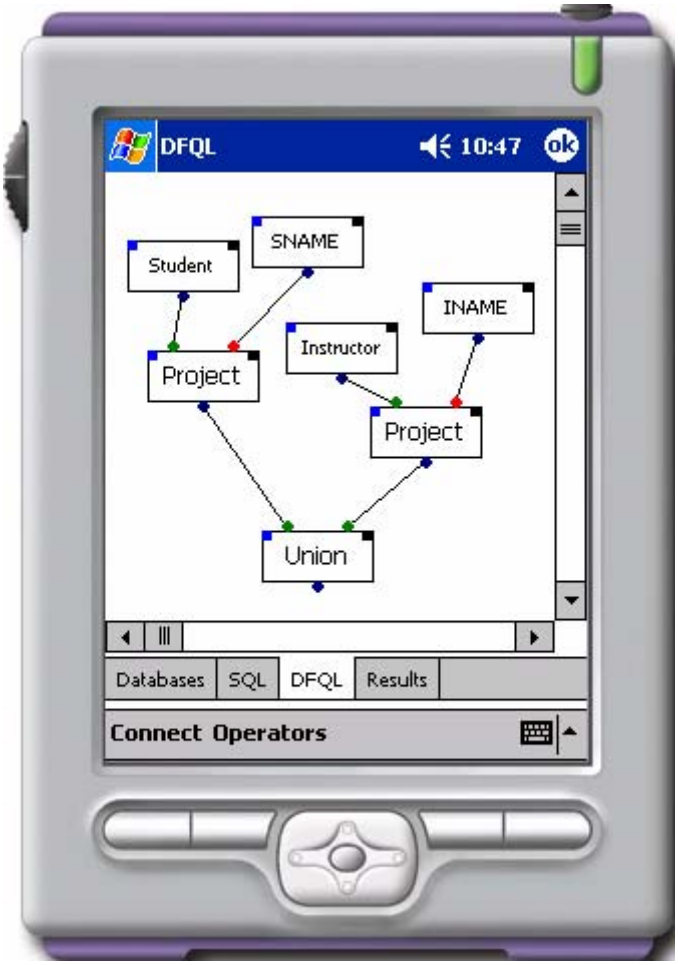


Figure 9. Union DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT sname FROM Student UNION SELECT DISTINCT iname FROM Instructor.*

7. Diff DFQL Operator

The user chooses the *Diff* operator from the *Basic* operators submenu and then taps on the screen for the *Diff* operator to appear. *Diff* operators have three input nodes. Two input nodes accept a *Relation*. The other input node accepts a *Criteria*. The user connects the input node by tapping on the input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

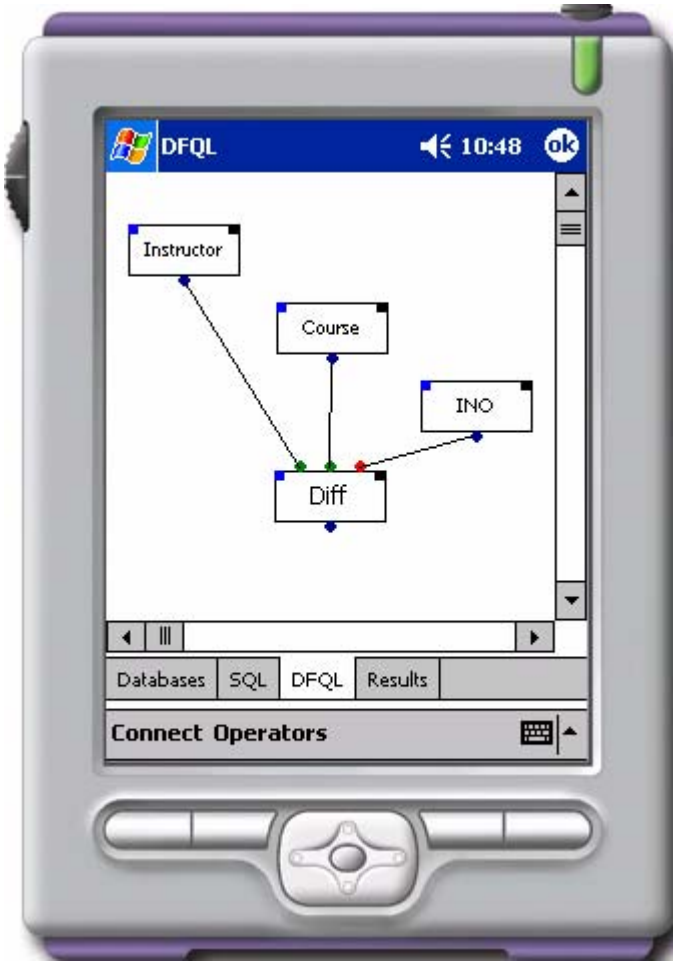


Figure 10. Diff DFQL Operator

The SQL translation of DFQL query above is: *SELECT DISTINCT ino FROM Instructor WHERE ino NOT IN (SELECT DISTINCT ino FROM COURSE.*

8. GrpCnt DFQL Operator

The user chooses the *GrpCnt* operator from the *Basic* operators submenu and then taps on the screen for the *GrpCnt* operator to appear. *GrpCnt* operators have three input nodes. One input node accepts a *Relation*. The other two input nodes accepts *Criterias*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

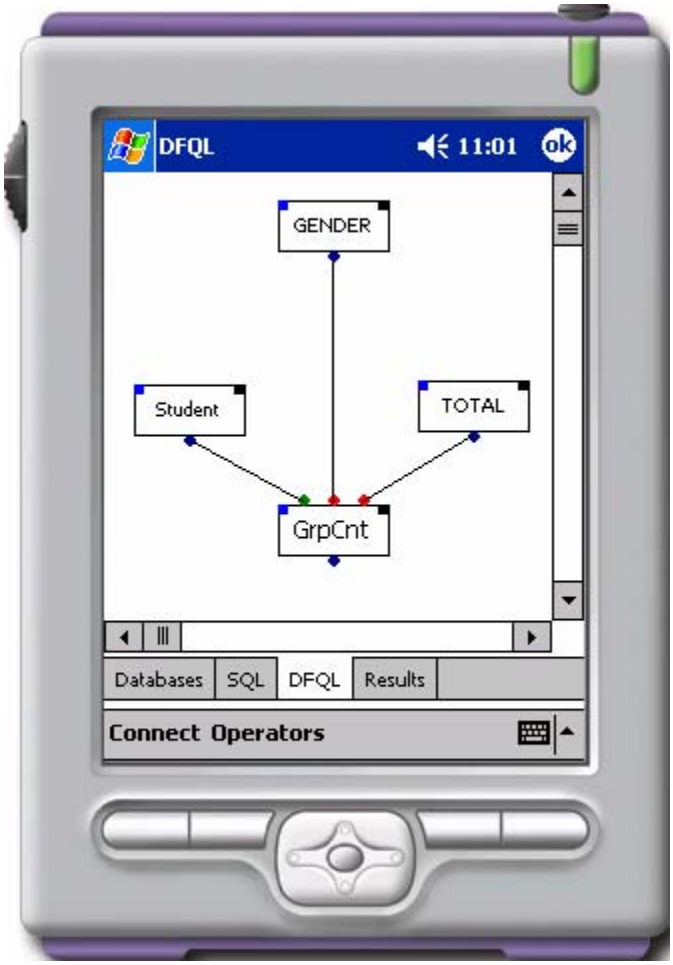


Figure 11. GrpCnt DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINTCT gender COUNT(*) total FROM Student GROUP BY gender.*

9. EqJoin DFQL Operator

The user chooses the *EqJoin* operator from the *Advanced* operators submenu and then taps on the screen for the *EqJoin* operator to appear. *EqJoin* operators have three input nodes. Two input nodes accepts *Relations*. The other input node accepts a *Criteria*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

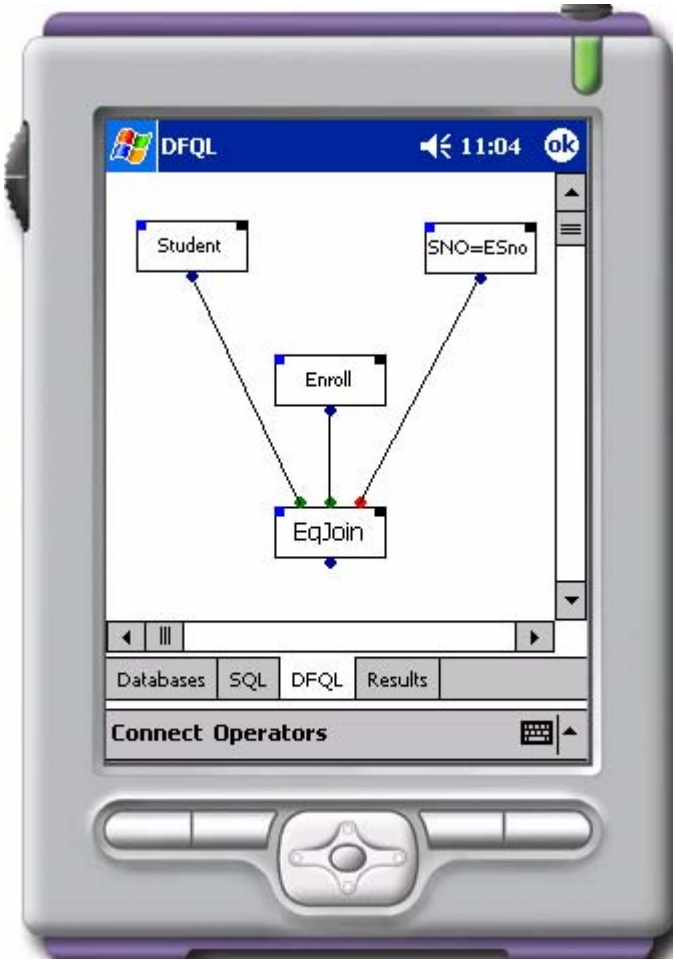


Figure 12. EqJoin DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINICT * FROM Student, Enroll WHERE Student.SNO = Enroll.Esno.*

10. GrpAllSat DFQL Operator

The user chooses the *GrpAllSat* operator from the *Advanced* operators submenu and then taps on the screen for the *GrpAllSat* operator to appear. *GrpAllSat* operators have three input nodes. One input node accepts a *Relation*. The other two input nodes accepts *Criterias*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

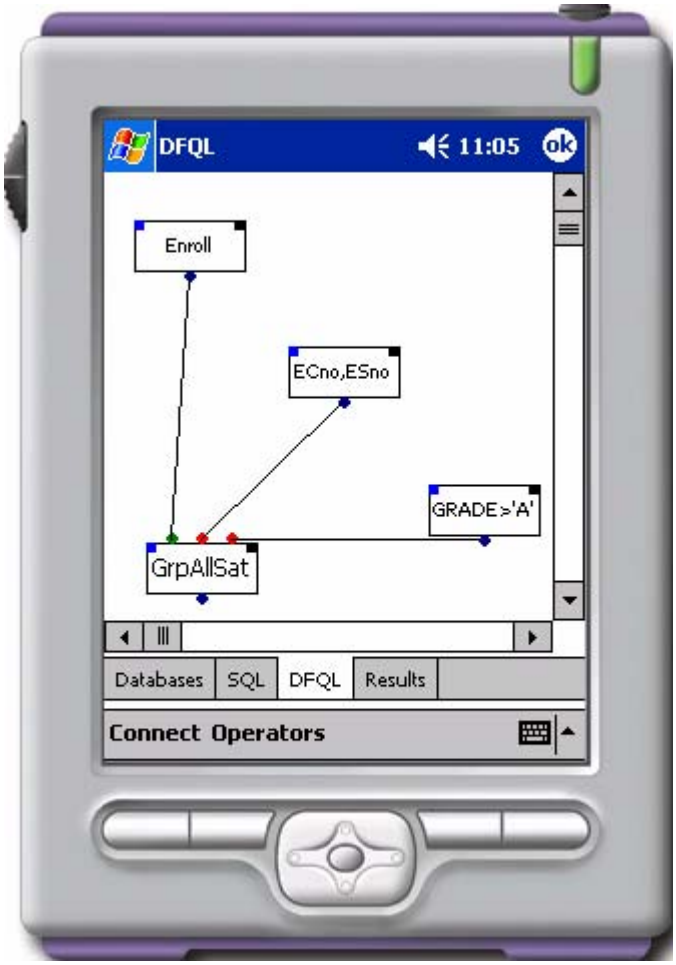


Figure 13. GrpAllSat DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT Ecno, Esno FROM Enroll WHERE grade>'a' GROUP BY Ecno, Esno.*

11. GrpNSat DFQL Operator

The user chooses the *GrpNSat* operator from the *Advanced* operators submenu and then taps on the screen for the *GrpNSat* operator to appear. *GrpNSat* operators have four input nodes. One input node accepts a *Relation*. The other three input nodes accept *Criterias*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

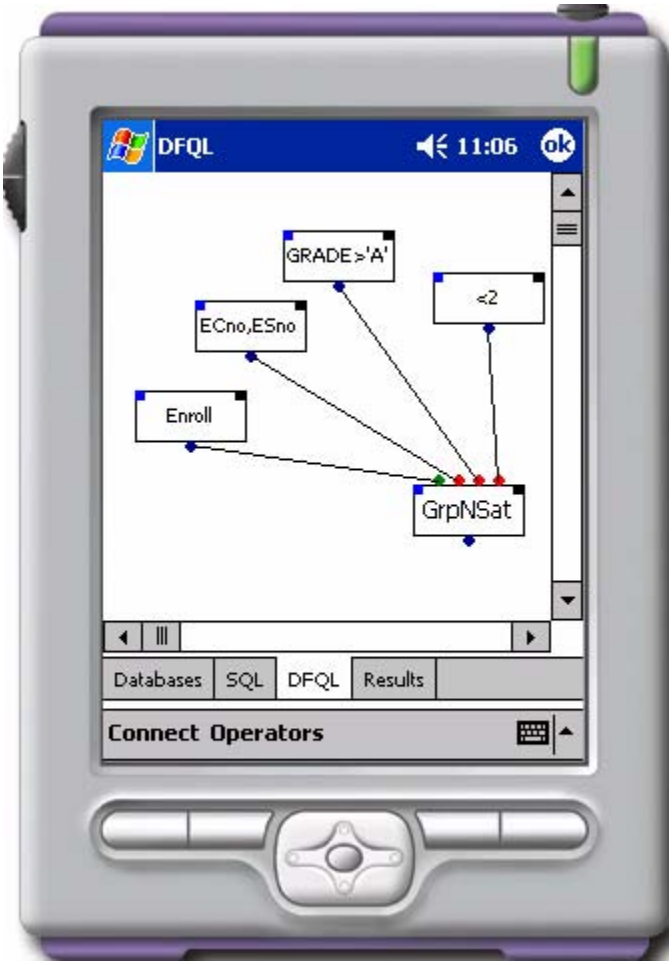


Figure 14. GrpNSat DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT Ecno, Esno FROM Enroll WHERE grade > 'a' GROUP BY Ecno, Esno HAVING COUNT (*) < 2.*

12. Intersect DFQL Operator

The user chooses the Intersect Operator from the Advanced Operators submenu and then taps on the screen for the Intersect Operator to appear. Intersect Operators have three input nodes. Two input nodes accepts a Relation. The other input node accepts Criteria. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

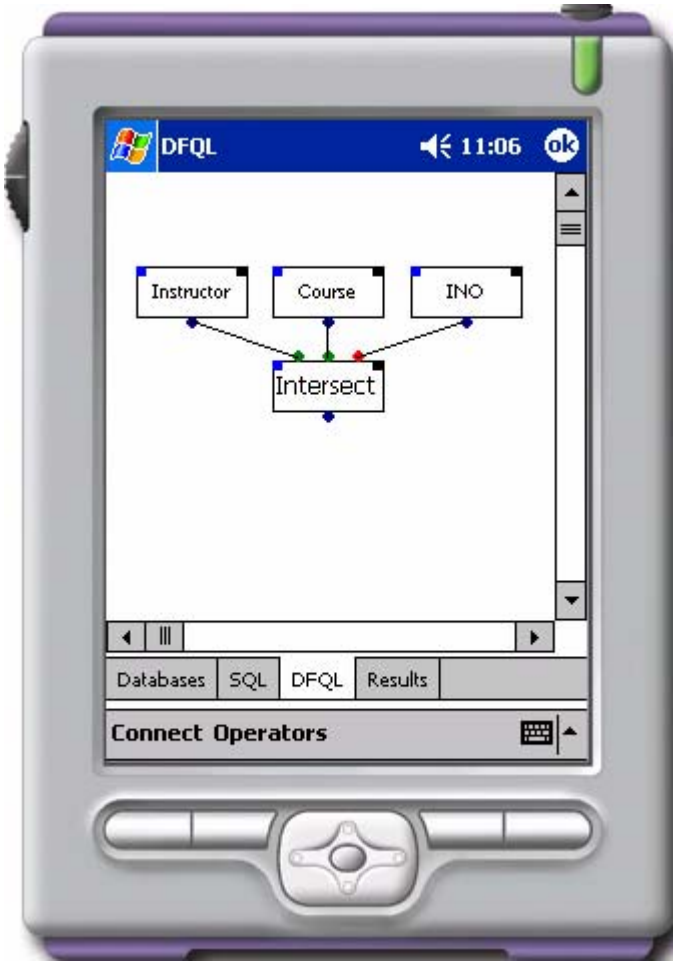


Figure 15. Intersect DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT ino FROM Instructor WHERE ino IN (SELECT ino FROM Course).*

13. GrpMin DFQL Operator

The user chooses the *GrpMin* operator from the *Advanced* operators submenu and then taps on the screen for the *GrpMin* operator to appear. *GrpMin* operators have three input nodes. One input node accepts a *Relation*. The other two input nodes accept *Criterias*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

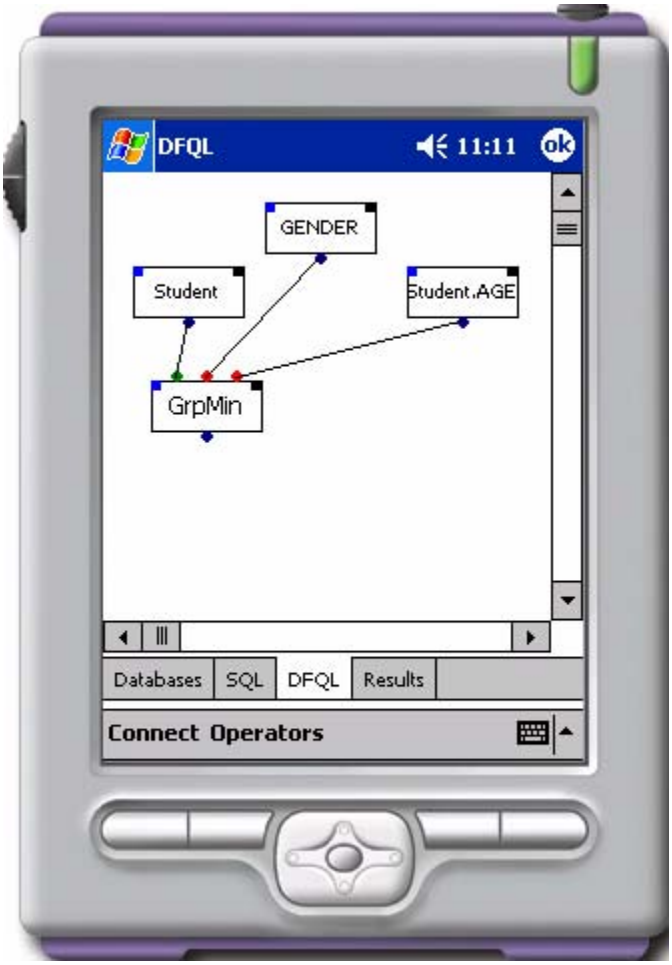


Figure 16. GrpMin DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT gender MIN (Student.Age) FROM Student GROUP BY gender*

14. GrpMax DFQL Operator

The user chooses the *GrpMax* operator from the *Advanced* operators submenu and then taps on the screen for the *GrpMax* operator to appear. *GrpMax* operators have three input nodes. One input node accepts a *Relation*. The other two input nodes accept *Criterias*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

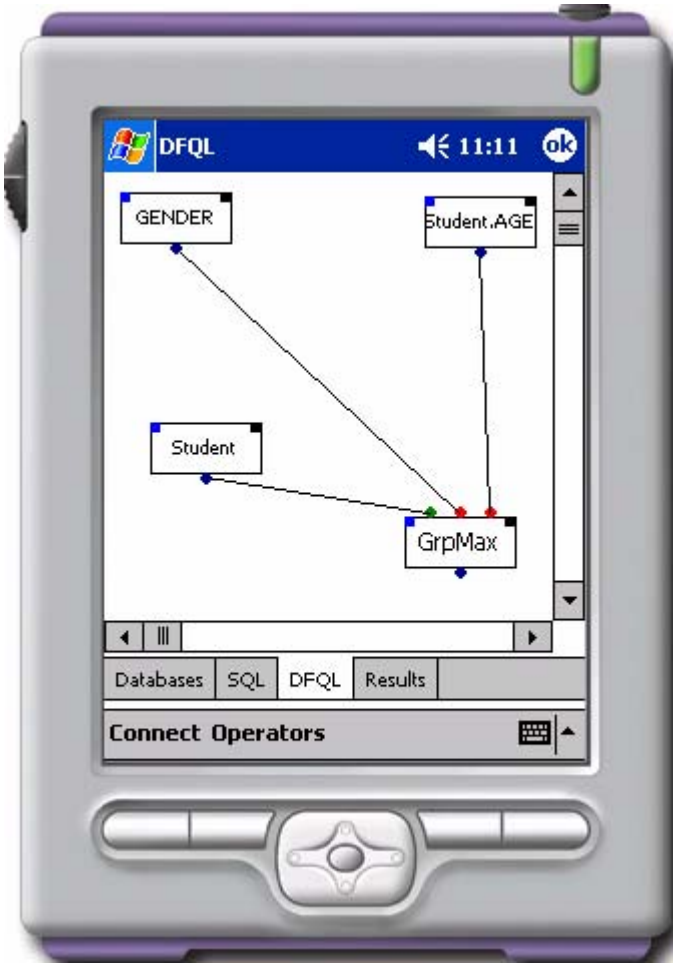


Figure 17. GrpMax DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT gender MAX (Student.Age) FROM Student GROUP BY gender*

15. GrpAvg DFQL Operator

The user chooses the *GrpAvg* operator from the *Advanced* operators submenu and then taps on the screen for the *GrpAvg* operator to appear. *GrpAvg* operators have three input nodes. One input node accepts a *Relation*. The two other input nodes accept *Criterias*. The user connects the input node by tapping on an input node and then the body of another operator. This creates a line that connects the input node to an output node of another operator.

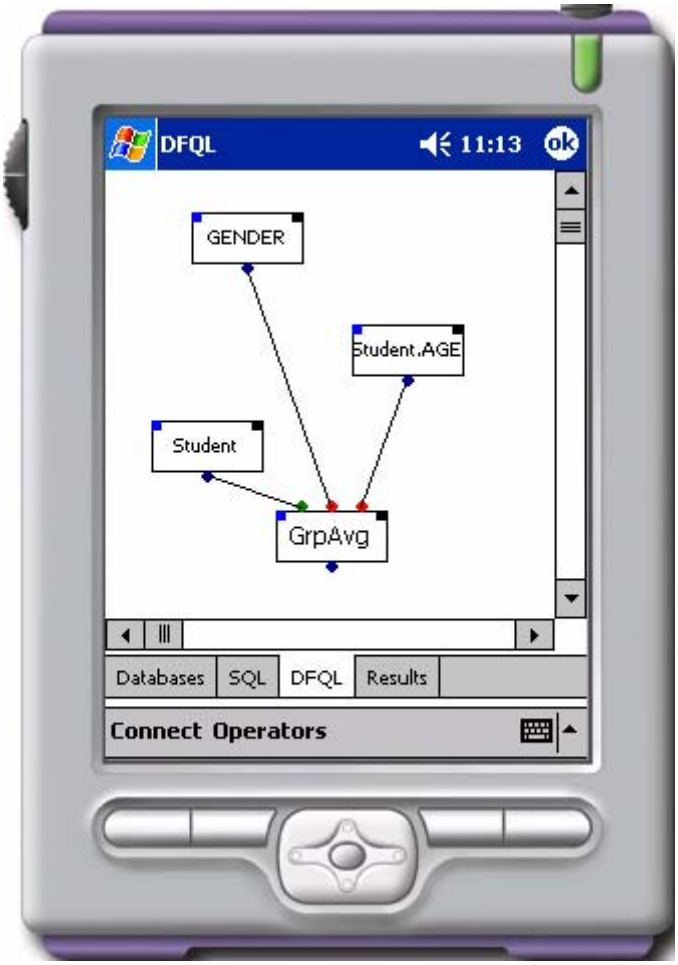


Figure 18. GrpAvg DFQL Operator

The SQL translation of the DFQL query above is: *SELECT DISTINCT gender AVG (Student.Age) FROM Student GROUP BY gender*

16. Incremental Queries

Below is an example of a complex query created using both SQL and DFQL for comparison.

English Query: Names of students that received A's on all courses taken.

SQL: SELECT sname FROM student WHERE sno IN (SELECT esno FROM enroll WHERE grade='a' AND esno NOT IN (SELECT esno FROM enroll WHERE grade <>'a'))

DFQL:

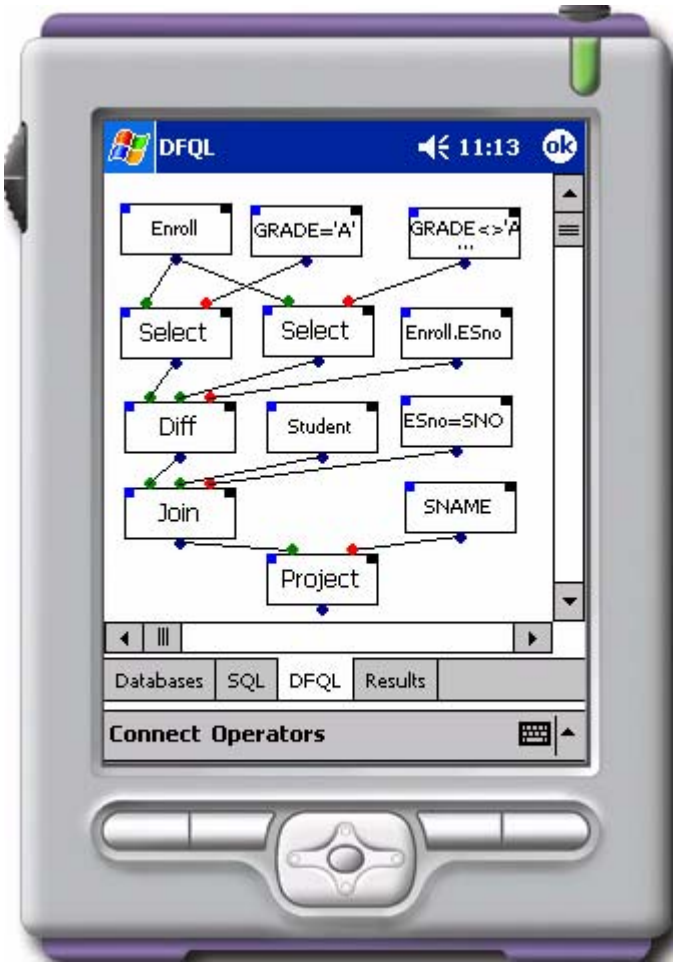


Figure 19. Incremental DFQL Query

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN

A. PLATFORM

As discussed earlier, cell phones, Blackberrys™, and PDAs are considered handheld devices. These handheld devices were investigated as platforms for implemention of the DFQL application. They were considered because they are all capable of network connectivity, a needed feature in order to access remote databases. Another capability these devices posses, is their ability display and accept information from user input. These is done with the use of display screen and user input devices.

1. Cell Phones

Cell phones have been in existence for more than a decade and have evolved tremendouosly. They've evolved from large cell phones the size of a shoe box to small flip phones that can fit in your pocket. Nowadays, some cell phones are Java enabled capable of storing different applications (i.e., games, music, photos). Cell phones are also capable of sending and receiving both e-mail and SMS messages. Finally, they can surf the web for online published information and file downloads.

Implementing the DFQL application, cell phones proved to be too small. The screen size does not have enough real estate to make it practical for DFQL. DFQL is a graphical application where icons and text information needed to be viewed in a large enough area that gives the user a visual perspective of the query. Creating a DFQL query on a cell phone where 3 or 4 DFQL icon operators would be a challenge to display went against the purpose of its creation.

Another short coming of using the cell phone platform was the availability of an easy to use user input device. Most cell phones use navigational buttons that can move a cursor up, down, left, right. Moving DFQL operators or connecting them would require tedious button interaction.

Figure 20. Cell Phone

2. Blackberry™

Blackberry™ devices have bigger screens than cell phones. It also has a builtin keyboard, which makes text entry easier than with cell phone type keypads. The wheel device on Blackberrys™ are used for scrolling and adds more efficiency than navigation buttons on cell phones. However, it lacks a pointing device, which is less convenient to relocate DFQL operator icons.

Figure 21. Blackberry™

3. Personal Digital Assistant (PDA)

There are many types of PDAs out in the market. Nevertheless, majority of PDAs are either running Palm™ OS or Windows CE™ that Pocket PC™ runs under. Certain models are capable of 802.11 network connectivity. At the same time, all PDAs have on screen keyboards and touch pens. Touch pens are user input devices that can be used much like a mouse. They can be used to drag and drop DFQL operator icons. They can also be used to click on menus and submenus. Additionally, PDAs also have display screens with enough real estate to fit more than 10 DFQL operator icons. By implementing horizontal and vertical scroll bars, the application can fit even more DFQL operator icons. PDAs can either have color or black and white displays. Therefore, all these features, compared to the short comings that other platforms have, make PDAs the ideal candidate for implementing DFQL.

PALM™ POCKET PC™

Figure 22. PDA

B. PROGRAMMING LANGUAGE

Previous DFQL development used the Java™ language thus it was considered initially. Java™ has the Java 2 Micro Edition (J2ME)™. J2ME™ is the scaled down version of Java™ 2 and is primarily created for mobile devices and other embedded devices [Ref. 7].

Another attractive feature of Java™ is its promise of being platform independent because of the use of the Java™ Virtual Machine. Programs

created using J2ME™ are called MIDlets. Both the Palm™ and Pocket PC™ have their own Mobile Information Development Profile (MIDP) that needs to be installed before being able to run MIDlets [Ref.7].

An initial application was created for the Palm™ platform for investigating its capabilities. From the initial tests it was discovered that MIDP did not have native support for horizontal scroll and grid tables. Native support for these controls would give results of database queries a cleaner look. Furthermore, the graphic support is limited, there is no native tree view class that can be used for viewing the database structure.

Figure 23. Palm™ Query Result

The plan of developing an application that can both run on the Palm™ and Pocket PC™ was discarded. Although possible, the language had minimal native support due to the fact that it was created more for mobile cell phones [Ref. 7].

Java™ can also be implemented on the Pocket PC™ using Personal Java. Neverthess, the Personal Java implementation for the Pocket PC™ has its own

short comings. It has no native support for a tree view class. Realizing that using Java might not be the best option, it was decided to look at other programming languages. C Sharp (C#™) is similar to Java™ [Ref. 3]. The transition from one programming language to another would not be too difficult. Microsoft™ has the .NET™ Compact Framework, the core programming interface for mobile devices that includes Windows CE™ platforms such as the Pocket PC™. Windows CE™ has a more capable graphics engine. It supports tree view class, table grid, and horizontal scroll [Ref. 6]. It was therefore decided to use .NET™ Compact Framework using C#™.

Tree View Horizontal Scroll and Grid

Figure 24. Pocket PC™ Supported Controls

C. REMOTE ACCESS IMPLEMENTATION

1. Options

There are four ways to access remote databases. Some are simple and require minimal configuration. While others require installation and configuration of other enterprise applications (i.e., web server, database servers) [Ref. 6]. Adding additional components to the total architecture adds to the complexity of both the application and the enterprise.

a. ActiveSync

The simplest way to transfer data to and from a Pocket PC™ device is with File Replication using *ActiveSync*. Pocket PC™ connects to a desktop computer using a cradle in order to synchronize the Personal Information Management data. In order to transfer database information using *ActiveSync*, the host database data is converted to XML files. Once in XML format, user can synchronize with handheld device and read data.

However, the mobile device is bound to a specific desktop host for any new data updates. Also, mobile device must always be connected to get the most updated data [Ref. 5].

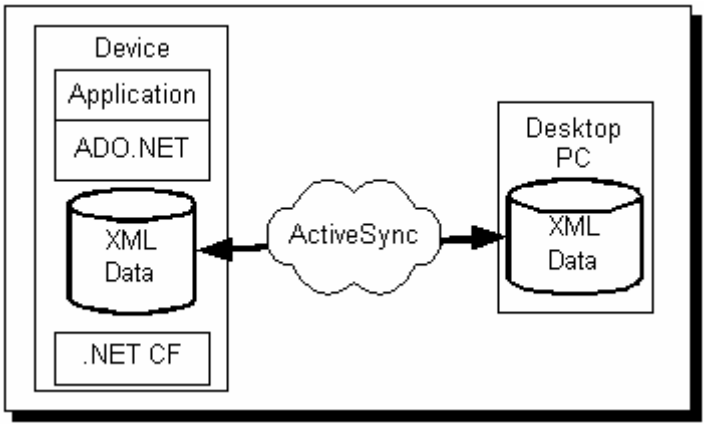


Figure 25. ActiveSync [Ref. 8]

b. Web Service

XML Web services are distributed software components accessible using standard Web protocols such as Simple Object Access Protocol (SOAP). A Web service comprises one or more Web methods, which a client can invoke to perform some application defined function. Data is returned from the Web method as an XML stream over HTTP. However, bandwidth can become an issue if Web methods take or return large amount of data because it tends to have a large overhead [Ref. 5].

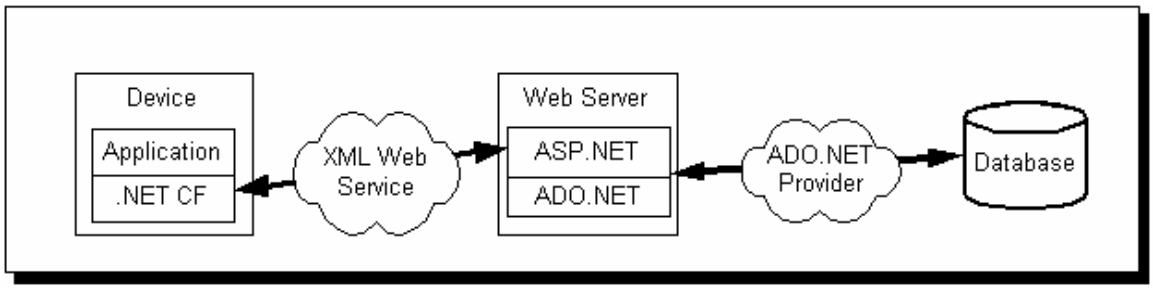


Figure 26. Web Service [Ref. 8]

c. SQL Server CE™

SQL Server CE™ is a compact relational database that runs on Windows CE™ devices. It was first released in 2000 and was already popular among Windows CE® developers prior to the appearance of the .NET™ Compact Framework. It is upwardly compatible with SQL Server, using compatible data types, and it has a small footprint, which is suitable for constrained devices [Ref. 5 pp 15-16]. It has built-in support for sharing data between a locally stored database and a remote SQL Server 2000™ database. By itself, SQL Server CE™ is a single-user database engine. However, at the same time, it can participate in ways to share a central database with other users through the support of the server-side database engine, SQL Server 2000™.

The .NET™ Compact Framework provides the following four capabilities for synchronizing mobile data with a SQL Server 2000™ database:

- Retrieve data from a SQL Server database into a SQL Server CE™ database on a device
- Add to, remove, or modify data at the device and have SQL Server CE™ track changes
- Have SQL Server CE™ submit the changed data to the SQL Server and receive any exceptions result from failed updates on the server
- Submit SQL statements directly from the .NET Compact Framework application to SQL Server database

The .NET Compact Framework, when combined with Windows CE™ and SQL Server, provides two methods for performing data transfer: Remote Data Access and Merge Replication.

(1) SQL Server CE™ with Remote Data Access (RDA) RDA provides all four capabilities listed above. It is Internet based and communicates between SQL Server CE™ and SQL Server 2000™ over LANs and WANs.

RDA supplies the functionality necessary for pulling data from an enterprise SQL Server 2000 database and loading it into tables held in a SQL Server CE™ database on the device. To save network bandwidth, data is compressed as it is transferred.

RDA is simple to set up and to use, but its primary disadvantage is that there is no conflict resolution. A developer must implement additional logic on the server to avoid overwriting updates from one client with those from another.

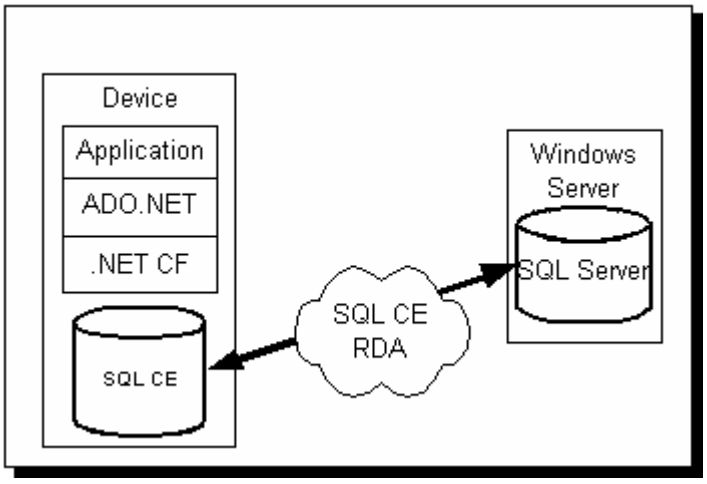


Figure 27. SQL Server CE™ with RDA [Ref. 8]

(2) SQL Server CE™ with Merge Replication. SQL Server CE™ with Merge Replication does not support direct submission of SQL statements. However, it provides the other three capabilities listed above [Ref. 6 pp 902-903].

SQL Server CE™ replication offers good support for wireless clients. As with RDA, replication compresses data transfers to conserve

bandwidth on the wireless network connection. If a connection is lost, the transfer resumes at the point at which it was cut off once the connection is reestablished.

An instance of SQL Server 2000™ executing in the enterprise can publish data that a SQL Server CE™ client can subscribe to. Data can be updated independently in the enterprise SQL Server 2000™ and on any subscribing client. Whenever a client resynchronizes, any updates made by the client are merged back into the master copy held in the enterprise SQL Server 2000™ database. Any changes made to the master copy since the data was last replicated are similarly merged into the SQL Server CE™ database on the mobile device, ensuring that the client copy remains up to date.

A SQL Server CE™ replication provider works with the SQL Server reconciler to manage replication and synchronization and to perform conflict resolution for SQL Server CE™ clients. When applications push updates back to the host server, either standard or custom conflict resolvers determine how to handle conflicting updates [Ref. 5 p 438].

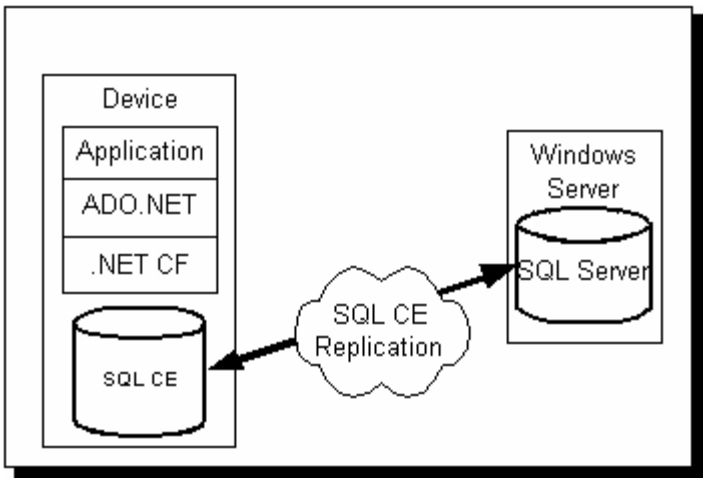


Figure 28. SQL Server CE™ with Replication [Ref. 8]

d. Direct Database Access

Direct access to SQL Server 2000™ provides fast access to the entire enterprise database without requiring the use of SQL Server CE™.

Queries can be very direct and focused to the needs of the application; the application accesses data only when absolutely required. However, the issue of memory requirements is becoming less with higher capacity devices and larger storage cards [Ref. 5 p 439].

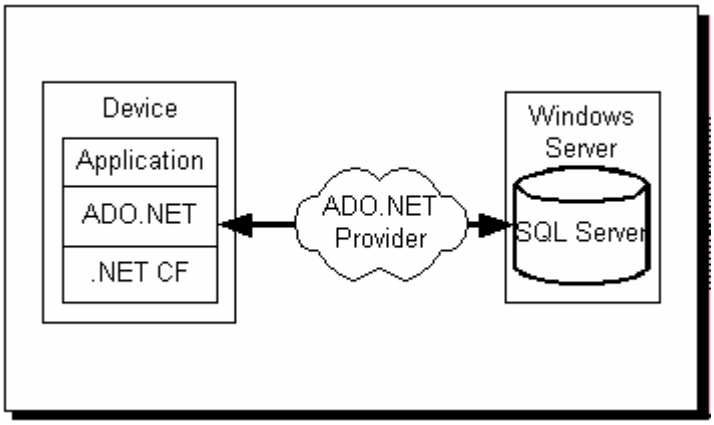


Figure 29. Direct Database Access [Ref. 8]

2. Requirements

Having requirements for the application helped determine which option was more suitable. One possible requirement is the ability of the application to handle any size database. If the application should download the whole database would the option chosen be able to handle the large amounts of data being received. Or would it be more efficient to query the database only when needed. However, for purposes of this thesis the application will not be subjected to extremely large databases. Creating one or finding one whose owners are willing to have an outside application queried against it would be outside the requirements of this thesis. It would be a welcome capability. However, it would lessen the complexity of the application without it.

Another possible requirement is with the issue of network connection. Does the application require continuous network connection or not? In order to answer this, another requirement needs to be addressed. Is there a requirement for the most up-to-date data? Should the application make constant access to the remote database in order to access the most up-to-date data or will a copy of

a not so old data suffice? For the purpose of this thesis, up-to-date data is not a necessity.

With regards to network bandwidth due to increased number of users, the application is targeted more towards technical users. Although DFQL is meant to make querying databases easier, it still requires the user to have knowledge of databases and SQL. It is designed to alleviate the complexities of SQL, easier but not necessarily easy. With this in mind, it is fair to assume that the number of users will not be so much as to bog down the network.

Lastly, DFQL is designed for querying databases. It was not designed for updating databases. Although having the ability to update databases using DFQL would be a welcome feature, it does not help alleviate the complexities of SQL. In this case, the issue of conflict resolution is not a requirement for the application.

After considering the different requirements and options available for implementation, it was decided to use RDA. RDA is best suited for mostly disconnected environment. Nevertheless, one might question why not choose Merge Replication instead. There is no requirement for SQL statements to be bounced against the remote database which RDA includes. At the same time, Merge Replication has conflict resolution functionality. However, conflict resolution is not a required functionality and most important RDA is simpler to set up.

RDA requires hard coding of which database and tables are to be downloaded. Rather than having a dynamic capability of choosing which database to download, the application developed for this thesis was hard coded to access a specific database and tables. A web service could have been created that published what databases are available and which tables the user wishes to download.

Additionally, Microsoft Internet Information Server™ (IIS) had to be configured in advance and the right permissions had to be set up for both IIS and Microsoft SQL Server 2000™. The database path for the local database had to

be determined in advance as well. Although a form could have been created where the user could be asked where the local copy of remote database should be saved. Nevertheless, for the purpose of this thesis, it was successfully shown that DFQL queries can access remote databases.

D. GUI MAIN DESIGN

The graphical user interface was designed keeping in mind that the application will be displayed in a small screen. The main challenge was how to display different information on a device with such a constraint. The application required ability to display connected database structure, SQL statements, DFQL queries, and query results. Each required its own window rather than sharing a small screen with one another. In order to accomplish this, each one was separated as tab pages.

A database tab was created solely for displaying connected database structure. The structure is displayed as a tree. The structure is displayed in such a way where tables are directly below the database node. While columns for each table where displayed in its respective table nodes.

Additionally, code was added in order to display data when user taps on either a table or column node. For example, when a user taps on a Table node named *Enroll* the application automatically activates the Results page tab and displays all rows for all columns found under the *Enroll* table. Also, the application writes the SQL equivalent in the SQL text box found in the SQL page tab. When the user taps on a column node, the same happens except that the application displays only all rows for the particular column of the table.

In order for the database tab to display a database structure, the user must first connect to a database either locally or by remote access. To do this, the user taps on the *Connect* menu item. This opens up a submenu with a local and remote menu item. If the user chooses a local database, the application opens up a load file form where all local databases are located. The user chooses a local database and clicks on the OK button. This will return the user back to the Databases tab page and display the structure of the chosen

database. On the other hand, should the user choose a remote database, the *Northwind* menu item shows up. This was hard coded into the application which downloads the whole *Northwind* user tables from a remote SQL Server 2000™ server connected by wireless connection and saves it on the handheld device as a local database called *Northwind*.

If the user connects to more additional databases, the database structure is appended to the end of the tree. The built in tree view control allows for expanding and collapsing of tree nodes. Also, the control supports horizontal and vertical scrolling and automatically activates when needed.

Figure 30. Database Tab Page

Previous implementation of DFQL had the capability of querying databases using SQL statements. This feature was adapted for this thesis as well. The SQL tab was created for this purpose. When the user clicks on the

SQL tab, the SQL tab page appears. Here a text box is available where the user can enter the SQL statement desired. The built-in text box control supports text wrapping and vertical scroll bars. This helps view the entire SQL statement in cases where the statements where very long. Once the desired SQL statement is entered the user clicks on the Run button. This will automatically bring up the Results tab page in order to display the result of the SQL statement.

The SQL tab page was designed in such a way so that when the Software Input Panel (SIP) was displayed to enter text information, which occupied about one third of the screen, it did not overlap over the Run button.

Figure 31. SQL Tab Page

The DFQL tab page is the center of the whole application. DFQL requires the maximum space the device could offer and at the same time still integrate a

mechanism to choose fifteen different DFQL operator icons. The implementation was finalized as having an Operator submenu for choosing an operator and then taping on the screen to display the operator at the chosen location. The DFQL tab page was also implemented with horizontal and vertical scroll bars. Although this did not give the application a whole view of the query, it gave the application a larger working space. With this implementation, operators can be outside the boundary of the screen and moved within viewing location by using the scroll bars.

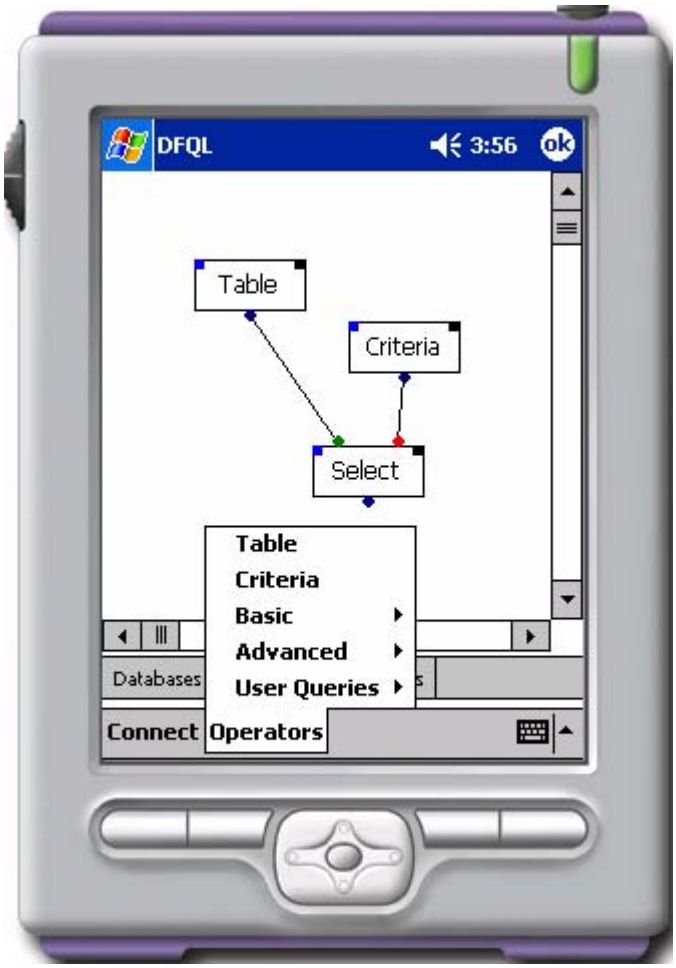


Figure 32. DFQL Tab Page

Finally, the Results tab page is used for displaying results of either an SQL or DFQL query. It was implemented by using a Data Grid control. Should the result be too large for the screen, a horizontal, vertical or both scroll bars

would appear making the result presented in a cleaner fashion. The page also automatically becomes active after either SQL or DFQL query is submitted.

Figure 33. Result Tab Page

E. CLASS DESIGN

The application can be subdivided into three main parts, graphical, database, and operator management.

Previously, the graphical part was discussed under the GUI Main Design section. Additionally, the graphical part has three classes: *Dfql*, *TableForm*, and *CriteriaForm*. Dfql displays the main program. The *TableForm* class is used when a *Table* operator body is tapped. This instantiats the class and displays a tree view of the different database tables available to choose from. Once a table node is chosen and the OK button tapped, the Table operator displays the table name. The *CriteriaForm* class has a similar function with the *TableForm* class.

However, instead of choosing just tables the *Criteria* class gave the user the ability to tap a node and have the information appear in a text box. As the user tapped more nodes, text is appended into the text box. This was designed to help the user minimize text typing with the SIP program. Instead, the user can use a combination of node taps and text typing to create a criteria. Once, the desired criteria is created, the user taps OK and the *Criteria* operator displays a short cut version of the text.

Next, the database management part comprise of the *AddLocalDatabase* class and *DatabaseUtility* class. The *AddLocalDatabase* class creates a local database every time the application is instantiated called *Enrollment.sdf*. The class initially creates tables and then columns. After which data are inserted into the database. The creation of a local database is needed in order to make sure that there is at least one local database that queries can be submitted against. On the other hand, the *DatabaseUtility* class is responsible for managing queries against databases. The application uses it to display database schema, get query results, and create temporary tables.

Finally, the operator management part is composed of the *Operator* class and *OperatorUtility* class. The *Operator* class is used to store data about an operator. Data stored are location, number of input nodes, operator name, input names, operator type, query statement, and location of database. The *OperatorUtility* class manages all operators created. *Operators* are stored as an array. The class takes care of adding, deleting, moving, and query processing. It also determines whether an operator had been clicked at and at what location, which determines whether a node was chosen. Query processing is done by recursion until a child leaf operator is reached. Once leaf operator is reached it is used to fill parts of dependent operator's query statement. The result of the query is then stored in a temporary table. The result acts as input for its dependent operator and the cycle repeats until the root operator is reached.

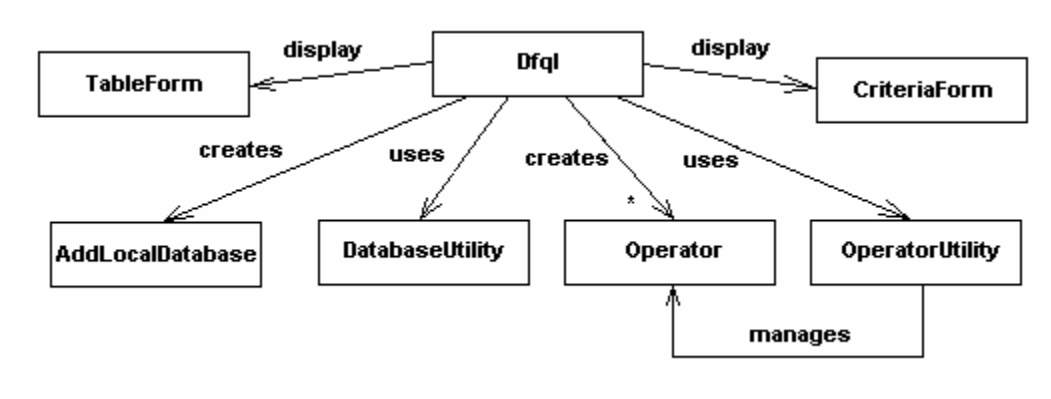


Figure 34. Class Diagram

IV. HARDWARE

Most testing was done using a Pocket PC™ emulator. Nevertheless, a physical hardware test configuration had to be built in order to test the final form of the application.

The laptop was an HP™ laptop with an Ethernet card. This laptop was wired into a Belkin™ 802.11b wireless router using twisted pair cable. The Pocket PC™ PDA was connected to the network using its internal 802.11b wireless device. With this configuration the laptop was connected to the network using a cable while the Pocket PC™ device connected wirelessly.

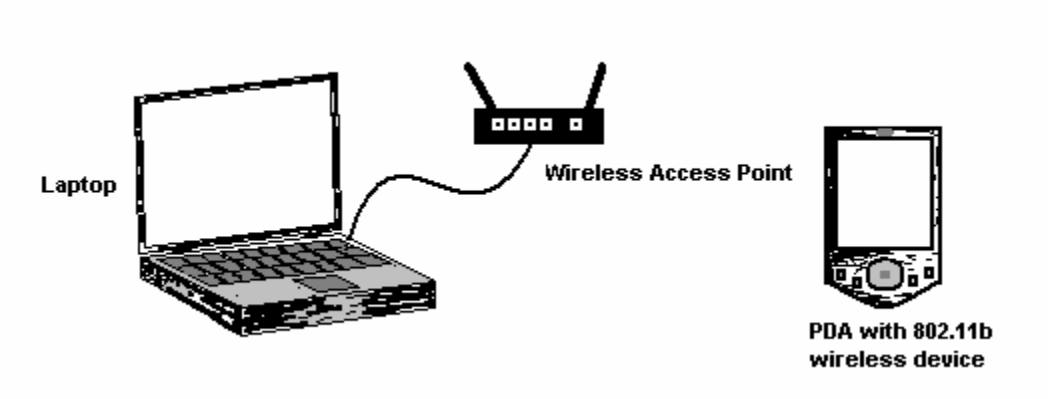


Figure 35. Hardware Configuration

The setup was problematic at first. The wireless access point had a built in firewall and prevented connection. In order to minimize complexity, the wireless access point was configured as an access point only, rather than a router/firewall. The laptop computer had Windows XP Professional™. It too had a built in firewall and had to be configured to allow port 80 inbound access. Once both the wireless access point and laptop was configured, the PDA was setup for wireless connection. However, due to multiple wireless access points that may be present in close proximity, one had to make sure the PDA was connecting to the right network.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SOFTWARE

The software development application used to develop the DFQL application was the Visual Studio .NET 2003™. Visual Studio .NET 2003™ supports .NET™ compact framework, it is the .NET development framework for smart devices. It supports three programming languages to develop smart device applications. They are: C#™, C++™, and Visual Basic™. The C#™ programming language was used to develop the DFQL application.

Several software were installed into the laptop. The operating system used on the laptop was initially a Windows XP Home Edition™. However, this version of XP does not include the Microsoft Internet Information Server™ (IIS) Web server software found in the Professional version. Therefore, the laptop had to be upgraded to Professional in order to install IIS. The Windows SQL Server 2000™ was also installed. SQL Server 2000™ had a sample database called Northwind. This database was used as the remote database for the DFQL application. Its permission had to be configured to allow anonymous access using the built in anonymous account for IIS. Another software installed was the SQL Server CE™ Server Agent. This software is used for creating and configuring the Web server to be accessed by client devices with SQL Server CE™. After installing Server Agent it still needs additional configuration by running the SQL Server CE™ Connectivity Management program. This program sets up the IIS Web server to establish a site for client device access. At the same time, this program sets up the appropriate permission for the created site. All these software were required to be installed in order to support remote database access capability for the DFQL application. Finally, the Microsoft™ ActiveSync was installed. This program is used to synchronize the Pocket PC™ device with the laptop using a USB connection. With this connection, any program created using Visual Studio™ on the laptop can be packaged and transferred to the Pocket PC™ device.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

DFQL had previously been implemented several times on a desktop machine. Finally, a PDA implementation was created giving the concept a wider implementation within an enterprise environment. DFQL helps alleviate the complexity of SQL. Its graphical interface helps visualize what happens to data as it passes through the different operators. An object oriented model in creating relational database queries. And even though PDAs have small screens, the application was implemented to be as powerful as its desktop counterpart.

The DFQL implementation for this thesis took previous shortcomings and implemented features to overcome it. Operators can be moved around, deleted, unconnected, or re-connected. Queries can be saved and re-loaded. The screen for creating DFQL queries can scroll vertically or horizontally and help overcome the screen size limitation. Another feature is that there is no need for display flags or display operators. Users just need to tap on an output node and the query will run from the chosen operator and results are displayed on a *Results* tab page. Output nodes can be tapped from the root node all the way to child nodes. This feature allows an easy way to view data rather than manually checking display flags or connecting display operators. At the same time, data held by Table and Criteria operators are displayable by packaging its data as data tables. This is done by design in order to view data in a much easier way. Most noteworthy, is the ability of the application to query separate databases and simultaneously combine its results for other operators to use. In essence, taking advantage of distributed databases and querying data as if it came from one large database.

Even though the programming language used is a scaled down version of the full .NET™ framework, the new implementation was still able to retain features found with earlier desktop implementations. Users are still capable of viewing database structure using a Tree structure. Data from Tables and Columns can be displayed by just tapping on the desired node. Also, query results are still

displayed in a table grid fashion. Finally, should the user want to create text SQL query that feature was retained as well.

The advantage of using a PDA is that the user is now mobile. While within range of a wireless network connection, user has the ability to access a remote database, download it, and then create DFQL queries against the local version. Having a local copy, the user is no longer constrained inside the confines of an office room. The user can now move outside the range of a wireless network and still be able to query and see results from its local copy of the database. Should the user have the need for the most up-to-date data, the user just moves within range of a wireless network and download the latest data.

One improvement that can be implemented would be with remote databases. Remote database access for this thesis is currently hard coded with regards to what database and what tables are to be downloaded. It would be better to use a combination of RDA and web service option. The web service would return to the connecting client information regarding what databases are available. After choosing a database the user can be given information as to what tables are available and how many rows it has. The user can then decide which tables to download. This type of implementation would have been a more dynamic design.

Also, previous desktop implementations had user-defined operators. User defined operators are created using built-in operators and helps minimize real estate consumption. The future mobile device implementation should include this feature for improved usability.

Nevertheless, DFQL has once again kept up to its promise. Querying relational databases is an easier process when using DFQL. Even when implemented on a device with a screen size limitation, DFQL can be implemented to be as powerful as its desktop counterparts.

APPENDIX – A

SOURCE CODE

1. Dfql.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Data.SqlServerCe;
using System.Text;
using System.Xml;
using System.IO;

namespace DFQL
{
        /**
         * Main class for DFQL application
         * Form class for user interface for connecting
         * to local and remote databases, creating DFQL,
         * and viewing results.
         * AUTHOR: Evangelista, Mark A.
        **/
        public class Dfql : System.Windows.Forms.Form
        {
                /**Main Menu of Application**/
                private System.Windows.Forms.MainMenu mainMenu;

                /**Sub menu of Main menu for connecting to databases**/
                privateSystem.Windows.Forms.MenuItem connectMenuItem;

                /**Sub menu of connectMenuItem for local databases**/
                private System.Windows.Forms.MenuItem localMenuItem;

                /**Sub menu of connectMenuItem for remote databases**/
                private System.Windows.Forms.MenuItem remoteMenuItem;

                /**Tab control of Application**/
                private System.Windows.Forms.TabControl tabControl;

                /**Tab page for viewing connected databases*/
```

```
private System.Windows.Forms.TabPage dbTab;

/**Built-in dialog for opening local file**/
private System.Windows.Forms.OpenFileDialog localFile;

/**Built-in dialog for saving file**/
private System.Windows.Forms.SaveFileDialog saveFile;

/**Tree view for displaying database schema**/
private System.Windows.Forms.TreeView treeView;

/**Tab page for viewing results of query**/
private System.Windows.Forms.TabPage resultTab;

/**Root node of tree for displaying database schema**/
private System.Windows.Forms.TreeNode rootNode;

/**XML test writer for saving database schema**/
private XmlTextWriter dbXml;

/**XML document for saving database schema**/
private XmlDocument xmlDoc;

/**Data grid for displaying query results in tabular form**/
private System.Windows.Forms.DataGrid resultGrid;

/**Text box to type SQL query for submission**/
private System.Windows.Forms.TextBox queryBox;

/**Button to run submitted SQL query**/
private System.Windows.Forms.Button runButton;

/**Image list for displaying different icons on tree**/
private System.Windows.Forms.ImageList iconImageList;

/**Tab page for creating and submitting SQL queries**/
private System.Windows.Forms.TabPage sqlTab;

/**Tab page for creating DFQL**/
private System.Windows.Forms.TabPage dfqlTab;

/**Label of SQL text box**/
private System.Windows.Forms.Label queryLabel;

/**Location of database**/
private string dbaseLoc = "";
```

```
/**SQL query**/
private string query = "";

/**Sub menu of Main menu for creating DFQL operators**/
private System.Windows.Forms.MenuItem operatorsMenuItem;

/**Sub menu of Operators menu for creating Basic DFQL
 * operators**/
private System.Windows.Forms.MenuItem basicMenuItem;

/**Sub menu of Basic menu for creating Select DFQL
 * operator**/
private System.Windows.Forms.MenuItem selectMenuItem;

/**Sub menu of Basic menu for creating Project DFQL
 * operator**/
private System.Windows.Forms.MenuItem projectMenuItem;

/**Sub menu of Basic menu for creating Join DFQL
 * operator**/
private System.Windows.Forms.MenuItem joinMenuItem;

/**Sub menu of Basic menu for creating Difference DFQL
 * operator**/
private System.Windows.Forms.MenuItem diffMenuItem;

/**Sub menu of Basic menu for creating Union DFQL
 * operator**/
private System.Windows.Forms.MenuItem unionMenuItem;

/**Sub menu of Basic menu for creating Group Count DFQL
 * operator**/
private System.Windows.Forms.MenuItem groupcntMenuItem;

/**Sub menu of Operators menu for creating Advanced DFQL
 * operators**/
private System.Windows.Forms.MenuItem advancedMenuItem;

/**Sub menu of Advanced menu for creating Eqjoin DFQL
 * operator**/
private System.Windows.Forms.MenuItem eqjoinMenuItem;

/**Sub menu of Advanced menu for creating Group All Satisfy
 * DFQL operator**/
private System.Windows.Forms.MenuItem grpallsatMenuItem;
```

```
/**Sub menu of Advanced menu for creating Group N Satisfy
 * DFQL operator**/
private System.Windows.Forms.MenuItem groupnsatMenuItem;

/**Sub menu of Advanced menu for creating Intersect DFQL
 * operator**/
private System.Windows.Forms.MenuItem intersectMenuItem;

/**Sub menu of Advanced menu for creating Group Minimum
 *  DFQL operator**/
private System.Windows.Forms.MenuItem groupminMenuItem;

/**Sub menu of Advanced menu for creating Group Maximum
 * DFQL operator**/
private System.Windows.Forms.MenuItem groupmaxMenuItem;

/**Sub menu of Advanced menu for creating Group Average
 * DFQL operator**/
private System.Windows.Forms.MenuItem groupavgMenuItem;

/**Sub menu of User Defined menu for loading user defined
 * DFQL operator**/
private System.Windows.Forms.MenuItem loadMenuItem;

/**Sub menu of User Defined menu for saving user defined
 * DFQL operator**/
private System.Windows.Forms.MenuItem saveMenuItem;

/**XML file location on mobile device**/
private static string xmfFilePath = @"My Documents\db.xml";

/**Action to be taken after a mouse down event**/
private string action;

/**Boolean flag to show operator information after mouse down
 * on body portion of operator**/
private bool show = true;

/**counter to be used to concatenate with operator type for
 * unique dynamic naming of each operator created**/
private int counter = 0;

/**Operator Utility class for keeping track of operators
 * created**/
private OperatorUtility ou;
```

```
/**Sub menu of Operators menu for creating Table DFQL operator**/
private System.Windows.Forms.MenuItem tableMenuItem;

/**Sub menu of Operators menu for creating Criteria DFQL operator**/
private System.Windows.Forms.MenuItem criteriaMenuItem;

/**Sub menu of Remote menu for connecting to remote Northwind database**/
private System.Windows.Forms.MenuItem menuRemoteNorthwind;

/**Horizontal Scroll Bar control for DFQL tab**/
private System.Windows.Forms.HScrollBar hScrollBar;

/**Vertical Scroll Bar control for DFQL tab**/
private System.Windows.Forms.VScrollBar vScrollBar;

/**Panel control used for creating scrollable panel**/
private System.Windows.Forms.Panel panelA;

/**Panel control used for creating displaying DFQL queries**/
private System.Windows.Forms.Panel panelB;

/**Sub menu of Operator menu for saving and loading user queries**/
private System.Windows.Forms.MenuItem usrQryMenuItem;

/**Operator class used for temporary storage of information before
 * mouse down event in order to know where to show operator on
 * screen before adding to list**/
private Operator op;

public Dfql()
{
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();
        moreInitialization();

        /**Create local database**/
        AddLocalDatabase localDb = new AddLocalDatabase();
```

```
        /**Event handler when tables or columns are clicked on
        tree**/
        treeView.AfterSelect+=new
        TreeViewEventHandler(colNodeQuery);
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
        base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        System.Resources.ResourceManager resources = new
        System.Resources.ResourceManager(typeof(Dfql));
        this.mainMenu = new System.Windows.Forms.MainMenu();
        this.connectMenuItem = new
        System.Windows.Forms.MenuItem();
        this.localMenuItem = new
        System.Windows.Forms.MenuItem();
        this.remoteMenuItem = new
        System.Windows.Forms.MenuItem();
        this.menuRemoteNorthwind = new
        System.Windows.Forms.MenuItem();
        this.operatorsMenuItem = new
        System.Windows.Forms.MenuItem();
        this.tableMenuItem = new
        System.Windows.Forms.MenuItem();
        this.criteriaMenuItem = new
        System.Windows.Forms.MenuItem();
        this.basicMenuItem = new
        System.Windows.Forms.MenuItem();
        this.selectMenuItem = new
        System.Windows.Forms.MenuItem();
        this.projectMenuItem = new
        System.Windows.Forms.MenuItem();
        this.joinMenuItem = new
        System.Windows.Forms.MenuItem();
```

```
this.diffMenuItem = new
System.Windows.Forms.MenuItem();
this.unionMenuItem = new
System.Windows.Forms.MenuItem();
this.groupcntMenuItem = new
System.Windows.Forms.MenuItem();
this.advancedMenuItem = new
System.Windows.Forms.MenuItem();
this.eqjoinMenuItem = new
System.Windows.Forms.MenuItem();
this.grpallsatMenuItem = new
System.Windows.Forms.MenuItem();
this.groupnsatMenuItem = new
System.Windows.Forms.MenuItem();
this.intersectMenuItem = new
System.Windows.Forms.MenuItem();
this.groupminMenuItem = new
System.Windows.Forms.MenuItem();
this.groupmaxMenuItem = new
System.Windows.Forms.MenuItem();
this.groupavgMenuItem = new
System.Windows.Forms.MenuItem();
this.usrQryMenuItem = new
System.Windows.Forms.MenuItem();
this.loadMenuItem = new
System.Windows.Forms.MenuItem();
this.saveMenuItem = new
System.Windows.Forms.MenuItem();
this.tabControl = new System.Windows.Forms.TabControl();
this.dbTab = new System.Windows.Forms.TabPage();
this.treeView = new System.Windows.Forms.TreeView();
this.iconImageList = new
System.Windows.Forms.ImageList();
this.sqlTab = new System.Windows.Forms.TabPage();
this.queryLabel = new System.Windows.Forms.Label();
this.runButton = new System.Windows.Forms.Button();
this.queryBox = new System.Windows.Forms.TextBox();
this.dfqlTab = new System.Windows.Forms.TabPage();
this.panelA = new System.Windows.Forms.Panel();
this.panelB = new System.Windows.Forms.Panel();
this.vScrollBar = new System.Windows.Forms.VScrollBar();
this.hScrollBar = new System.Windows.Forms.HScrollBar();
this.resultTab = new System.Windows.Forms.TabPage();
this.resultGrid = new System.Windows.Forms.DataGrid();
this.localFile = new
System.Windows.Forms.OpenFileDialog();
```

```
this.saveFile = new System.Windows.Forms.SaveFileDialog();

//
// mainMenu
//
this.mainMenu.MenuItems.Add(this.connectMenuItem);
this.mainMenu.MenuItems.Add(this.operatorsMenuItem);

//
// connectMenuItem
//
this.connectMenuItem.MenuItems.Add(this.localMenuItem);
this.connectMenuItem.MenuItems.Add
(this.remoteMenuItem);
this.connectMenuItem.Text = "Connect";
this.connectMenuItem.Popup += new System.EventHandler(this.connectMenuItem_Popup);

//
// localMenuItem
//
this.localMenuItem.Text = "Local";
this.localMenuItem.Click += new System.EventHandler(this.local_Click);

//
// remoteMenuItem
//
this.remoteMenuItem.MenuItems.Add
(this.menuRemoteNorthwind);
this.remoteMenuItem.Text = "Remote";

//
// menuRemoteNorthwind
//
this.menuRemoteNorthwind.Text = "Northwind";
this.menuRemoteNorthwind.Click += new System.EventHandler(this.menuRemoteNorthwind_Click);

//
// operatorsMenuItem
//
this.operatorsMenuItem.MenuItems.Add
(this.tableMenuItem);
```

```
this.operatorsMenuItem.MenuItems.Add
(this.criteriaMenuItem);
this.operatorsMenuItem.MenuItems.Add
(this.basicMenuItem);
this.operatorsMenuItem.MenuItems.Add
(this.advancedMenuItem);
this.operatorsMenuItem.MenuItems.Add
(this.usrQryMenuItem);
this.operatorsMenuItem.Text = "Operators";
this.operatorsMenuItem.Popup += new
System.EventHandler(this.operatorsMenuItem_Popup);

//
// tableMenuItem
//
this.tableMenuItem.Text = "Table";
this.tableMenuItem.Click += new
System.EventHandler(this.tableMenuItem_Click);

//
// criteriaMenuItem
//
this.criteriaMenuItem.Text = "Criteria";
this.criteriaMenuItem.Click += new
System.EventHandler(this.criteriaMenuItem_Click);

//
// basicMenuItem
//
this.basicMenuItem.MenuItems.Add(this.selectMenuItem);
this.basicMenuItem.MenuItems.Add(this.projectMenuItem);
this.basicMenuItem.MenuItems.Add(this.joinMenuItem);
this.basicMenuItem.MenuItems.Add(this.diffMenuItem);
this.basicMenuItem.MenuItems.Add(this.unionMenuItem);
this.basicMenuItem.MenuItems.Add
(this.groupcntMenuItem);
this.basicMenuItem.Text = "Basic";

//
// selectMenuItem
//
this.selectMenuItem.Text = "SELECT";
this.selectMenuItem.Click += new
System.EventHandler(this.selectMenuItem_Click);

//
```

```
// projectMenuItem
//
this.projectMenuItem.Text = "PROJECT";
this.projectMenuItem.Click += new
System.EventHandler(this.projectMenuItem_Click);

//
// joinMenuItem
//
this.joinMenuItem.Text = "JOIN";
this.joinMenuItem.Click += new
System.EventHandler(this.joinMenuItem_Click);

//
// diffMenuItem
//
this.diffMenuItem.Text = "DIFF";
this.diffMenuItem.Click += new
System.EventHandler(this.diffMenuItem_Click);

//
// unionMenuItem
//
this.unionMenuItem.Text = "UNION";
this.unionMenuItem.Click += new
System.EventHandler(this.unionMenuItem_Click);

//
// groupcntMenuItem
//
this.groupcntMenuItem.Text = "GROUPCNT";
this.groupcntMenuItem.Click += new
System.EventHandler(this.groupcntMenuItem_Click);

//
// advancedMenuItem
//
this.advancedMenuItem.MenuItems.Add
(this.eqjoinMenuItem);
this.advancedMenuItem.MenuItems.Add
(this.grpallsatMenuItem);
this.advancedMenuItem.MenuItems.Add
(this.groupnsatMenuItem);
this.advancedMenuItem.MenuItems.Add
(this.intersectMenuItem);
this.advancedMenuItem.MenuItems.Add
```

```
(this.groupminMenuItem);
this.advancedMenuItem.MenuItems.Add
(this.groupmaxMenuItem);
this.advancedMenuItem.MenuItems.Add
(this.groupavgMenuItem);
this.advancedMenuItem.Text = "Advanced";

//
// eqjoinMenuItem
//
this.eqjoinMenuItem.Text = "EQJOIN";
this.eqjoinMenuItem.Click += new
System.EventHandler(this.eqjoinMenuItem_Click);

//
// grpallsatMenuItem
//
this.grpallsatMenuItem.Text = "GRPALLSAT";
this.grpallsatMenuItem.Click += new
System.EventHandler(this.grpallsatMenuItem_Click);

//
// groupnsatMenuItem
//
this.groupnsatMenuItem.Text = "GROUPNSAT";
this.groupnsatMenuItem.Click += new
System.EventHandler(this.groupnsatMenuItem_Click);

//
// intersectMenuItem
//
this.intersectMenuItem.Text = "INTERSECT";
this.intersectMenuItem.Click += new
System.EventHandler(this.intersectMenuItem_Click);

//
// groupminMenuItem
//
this.groupminMenuItem.Text = "GROUPMIN";
this.groupminMenuItem.Click += new
System.EventHandler(this.groupminMenuItem_Click);

//
// groupmaxMenuItem
//
this.groupmaxMenuItem.Text = "GROUPMAX";
```

```
this.groupmaxMenuItem.Click += new
System.EventHandler(this.groupmaxMenuItem_Click);

//
// groupavgMenuItem
//
this.groupavgMenuItem.Text = "GROUPAVG";
this.groupavgMenuItem.Click += new
System.EventHandler(this.groupavgMenuItem_Click);

//
// usrQryMenuItem
//
this.usrQryMenuItem.MenuItems.Add(this.loadMenuItem);
this.usrQryMenuItem.MenuItems.Add(this.saveMenuItem);
this.usrQryMenuItem.Text = "User Queries";

//
// loadMenuItem
//
this.loadMenuItem.Text = "Load";
this.loadMenuItem.Click += new
System.EventHandler(this.loadMenuItem_Click);

//
// saveMenuItem
//
this.saveMenuItem.Text = "Save";
this.saveMenuItem.Click += new
System.EventHandler(this.saveMenuItem_Click);

//
// tabControl
//
this.tabControl.Controls.Add(this.dbTab);
this.tabControl.Controls.Add(this.sqlTab);
this.tabControl.Controls.Add(this.dfqlTab);
this.tabControl.Controls.Add(this.resultTab);
this.tabControl.SelectedIndex = 0;
this.tabControl.Size = new System.Drawing.Size(240, 264);

//
// dbTab
//
this.dbTab.Controls.Add(this.treeView);
this.dbTab.Location = new System.Drawing.Point(4, 4);
```

```
this.dbTab.Size = new System.Drawing.Size(232, 238);
this.dbTab.Text = "Databases";

//
// treeView
//
this.treeView.ImageList = this.iconImageList;
this.treeView.Location = new System.Drawing.Point(8, 8);
this.treeView.Size = new System.Drawing.Size(216, 224);

//
// iconImageList
//
this.iconImageList.Images.Add(((System.Drawing.Image)(re
sources.GetObject("resource"))));
this.iconImageList.Images.Add(((System.Drawing.Image)(re
sources.GetObject("resource1"))));
this.iconImageList.Images.Add(((System.Drawing.Image)(re
sources.GetObject("resource2"))));
this.iconImageList.Images.Add(((System.Drawing.Image)(re
sources.GetObject("resource3"))));
this.iconImageList.ImageSize                =                new
System.Drawing.Size(16, 16);

//
// sqlTab
//
this.sqlTab.Controls.Add(this.queryLabel);
this.sqlTab.Controls.Add(this.runButton);
this.sqlTab.Controls.Add(this.queryBox);
this.sqlTab.Location = new System.Drawing.Point(4, 4);
this.sqlTab.Size = new System.Drawing.Size(232, 238);
this.sqlTab.Text = "SQL";

//
// queryLabel
//
this.queryLabel.Font = new System.Drawing.Font("Tahoma",
8.25F, System.Drawing.FontStyle.Bold);
this.queryLabel.Location = new System.Drawing.Point(8, 8);
this.queryLabel.Size = new System.Drawing.Size(216, 32);
this.queryLabel.Text = "SQL statement for: ";

//
// runButton
//
```

```
this.runButton.Location = new System.Drawing.Point(88,
160);
this.runButton.Size = new System.Drawing.Size(56, 24);
this.runButton.Text = "Run";
this.runButton.Click += new
System.EventHandler(this.runButton_Click);

//
// queryBox
//
this.queryBox.Font = new System.Drawing.Font("Tahoma",
8.25F, System.Drawing.FontStyle.Regular);
this.queryBox.Location = new System.Drawing.Point(8, 40);
this.queryBox.Multiline = true;
this.queryBox.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
this.queryBox.Size = new System.Drawing.Size(216, 112);
this.queryBox.Text = "";

//
// dfqlTab
//
this.dfqlTab.Controls.Add(this.panelA);
this.dfqlTab.Controls.Add(this.vScrollBar);
this.dfqlTab.Controls.Add(this.hScrollBar);
this.dfqlTab.Location = new System.Drawing.Point(4, 4);
this.dfqlTab.Size = new System.Drawing.Size(232, 238);
this.dfqlTab.Text = "DFQL";

//
// panelA
//
this.panelA.Controls.Add(this.panelB);
this.panelA.Size = new System.Drawing.Size(224, 224);

//
// panelB
//
this.panelB.Size = new System.Drawing.Size(432, 448);

//
// vScrollBar
//
this.vScrollBar.LargeChange = 20;
this.vScrollBar.Location = new System.Drawing.Point(224,
0);
```

```
this.vScrollBar.Maximum = 91;
this.vScrollBar.Size = new System.Drawing.Size(16, 224);
this.vScrollBar.SmallChange = 10;
this.vScrollBar.ValueChanged += new
System.EventHandler(this.vScrollBar_ValueChanged);

//
// hScrollBar
//
this.hScrollBar.LargeChange = 20;
this.hScrollBar.Location = new System.Drawing.Point(0,
224);
this.hScrollBar.Maximum = 91;
this.hScrollBar.Size = new System.Drawing.Size(224, 16);
this.hScrollBar.SmallChange = 10;
this.hScrollBar.ValueChanged += new
System.EventHandler(this.hScrollBar_ValueChanged);

//
// resultTab
//
this.resultTab.Controls.Add(this.resultGrid);
this.resultTab.Location = new System.Drawing.Point(4, 4);
this.resultTab.Size = new System.Drawing.Size(232, 238);
this.resultTab.Text = "Results";

//
// resultGrid
//
this.resultGrid.Location = new System.Drawing.Point(8, 8);
this.resultGrid.Size = new System.Drawing.Size(216, 224);
this.resultGrid.Text = "resultGrid";

//
// localFile
//
this.localFile.Filter = "SQLCE Databases(*.sdf)|*.sdf|All
Files(*.*)|*.*";
this.localFile.InitialDirectory = "\\My Documents";

//
// saveFile
//
this.saveFile.FileName = "Dfql1";
this.saveFile.Filter = "DFQL Queries(*.udo)|*.udo|All
Files(*.*)|*.*";
```

```
            this.saveFile.InitialDirectory = "\\My Documents";

            //
            // Dfql
            //
            this.Controls.Add(this.tabControl);
            this.Icon    =    ((System.Drawing.Icon)(resources.GetObject
            ("$this.Icon")));
            this.Menu = this.mainMenu;
            this.MinimizeBox = false;
            this.Text = "DFQL";

}
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>

static void Main()
{
            Application.Run(new Dfql());
}

/**
 * Calls builtin dialog after clicking local sub menu of
 * connect menu for choosing local database.  Creates tree
 * display of database chosen showing database name,
 * tables, and columns.  Also creates XML file for storing
 * database schema information.
 * **/
private void local_Click(object sender, System.EventArgs e)
{
            DialogResult dres = localFile.ShowDialog();
            if (dres == DialogResult.OK)
            {
                        createTreeView(localFile.FileName);
            }
}

/**
 * More initialization not captured by Windows Form
 * Designer generated code.
 * Initializes XML text writer, root tree node,
 * Paint, mouse down, mouse move, mouse up, and
 * Operator Utility.
```

```
**/
private void moreInitialization()
{
        this.dbXml = new XmlTextWriter(xmfFilePath,
        Encoding.ASCII);
        dbXml.Formatting = Formatting.Indented;
        dbXml.WriteStartDocument();
        dbXml.WriteStartElement("Databases");
        dbXml.WriteEndElement();
        dbXml.WriteEndDocument();
        dbXml.Close();

        StreamReader strRdr = new StreamReader(xmfFilePath);
        XmlTextReader xmlTxtRdr = new XmlTextReader(strRdr);
        while (xmlTxtRdr.Read())
        {
                if(xmlTxtRdr.NodeType.ToString() == "Element")
                {
                        if(xmlTxtRdr.Name == "Databases")
                        {
                                this.rootNode = new
                                System.Windows.Forms.TreeNode();
                                this.rootNode.Text = "Databases";
                                this.treeView.Nodes.Add(rootNode);
                        }
                }
        }
        xmlTxtRdr.Close();
        strRdr.Close();
        this.panelB.Paint += new PaintEventHandler(DFQL_Paint);
        this.panelB.MouseDown += new
        MouseEventHandler(DFQL_MouseDown);
        this.panelB.MouseMove += new
        MouseEventHandler(DFQL_MouseMove);
        this.panelB.MouseUp += new
        MouseEventHandler(DFQL_MouseUp);
        this.vScrollBar.Minimum = 0;
        this.hScrollBar.Minimum = 0;
        this.vScrollBar.Maximum = panelB.Height-panelA.Height;
        this.hScrollBar.Maximum = panelB.Width-panelA.Width;
        this.ou = new OperatorUtility();
        this.action = "none";
        op = new Operator();
}

/**
```

```
 * Captures SQL statement entered in text box, runs query,
 * and shows result in result tab page.
**/
private void runButton_Click(object sender, System.EventArgs e)
{
        this.query = this.queryBox.Text;
        DatabaseUtility runButtonDu = new DatabaseUtility();
        DataTable rslt = runButtonDu.queryDb(this.dbaseLoc,
        this.query);
        this.resultGrid.DataSource = rslt;
        this.tabControl.SelectedIndex = 3;

}

/**
 * Shows result of Table or Columns data in results
 * tab page.
**/
private void colNodeQuery(System.Object sender,
        System.Windows.Forms.TreeViewEventArgs e)
{
        int tableLoc = -1;

        try
        {
                tableLoc = e.Node.FullPath.IndexOf('\\',10);
        }
        catch
        {
        }

        if (tableLoc > -1)
        {
                this.dbaseLoc = @"\My Documents\" +
                e.Node.FullPath.Substring (10,tableLoc - 10) + ".sdf";
                int columnLoc = e.Node.FullPath.IndexOf
                ('\\',tableLoc + 1);

                DatabaseUtility colNodeQryDu = new
                DatabaseUtility();
                DataTable res = new DataTable();
                if (columnLoc > -1)
                {
                        string tblName = e.Node.Parent.Text;
                this.query = "Select " + e.Node.Text + " From " +
                tblName;
```

```
                }
                else
                {
                        this.query = "Select * From " + e.Node.Text;
                }
                res = colNodeQryDu.queryDb
                (this.dbaseLoc,this.query);
                this.resultGrid.DataSource = res;
                this.queryLabel.Text = "Enter SQL statement for: " +
                e.Node.FullPath.Substring(10,tableLoc - 10) + "
                database.";
                this.queryBox.Text = this.query;
                this.tabControl.SelectedIndex = 3;
        }
}

/**
 * Makes sure database tab page is displayed when connect
 * menu is selected
**/
private void connectMenuItem_Popup(object sender, EventArgs e)
{
        this.tabControl.SelectedIndex = 0;
}

/**
 * Makes sure operator tab page is displayed when operator
 * menu is selected
**/
private void operatorsMenuItem_Popup(object sender,
System.EventArgs e)
{
        this.tabControl.SelectedIndex = 2;
}

/**
 * Determines what to do during mouse down.
**/
private void DFQL_MouseDown(object sender, MouseEventArgs e)
{

        Point newDown = new Point(0,0);
        newDown.X = e.X;
        newDown.Y = e.Y;

        /**Create an operator at mouse down location**/
```

```
if (action == "create")
{
	if (ou.clickedAt(newDown) == "space")
	{
		op.setX(e.X);
		op.setY(e.Y);
		ou.add(op);
		counter++;
	}
}

/**
 * Updates data for input node A with name of connected
 * operator
**/
if (action == "connectA")
{
	Operator opTemp = ou.getOperatorTemp();
	action = ou.clickedAt(newDown);
	if (action != "move")
	{
		string [] input = opTemp.getInput();
		if (action == "output" || action == "body")
		{
			input [0] =
			ou.getOperatorTemp().getName();
			if (opTemp.getName().Equals(input [0]))
			{
				input [0] = "inputA";
			}
		}
		else
		{
			input [0] = "inputA";
		}
	}
	else
	{
		ou.add();
	}
	this.show = false;
}

/**
 * Updates data for input node B with name of connected
 * operator
```

```
**/
if (action == "connectB")
{
	Operator opTemp = ou.getOperatorTemp();
	action = ou.clickedAt(newDown);
	string [] input = opTemp.getInput();
	if(action != "move")
	{
		if (opTemp.getNumberOfInputNodes() == 4)
		{
			if (action == "output" || action == "body")
			{
				input [1] = ou.getOperatorTemp().getName();
				if (opTemp.getName().Equals(input [1]))
				{
					input [1] = "inputB";
				}
			}
			else
			{
				input [1] = "inputB";
			}
		}
	}
	else
	{
		ou.add();
	}
	this.show = false;
}

/**
 * Updates data for input node D with name of connected
 * operator
**/
if (action == "connectD")
{
	Operator opTemp = ou.getOperatorTemp();
	action = ou.clickedAt(newDown);
	string [] input = opTemp.getInput();
	if (action != "move")
	{
```

```
                    if (opTemp.getNumberOfInputNodes() == 4)
                    {
                        if (action == "output" || action == "body")
                        {
                            input [2] =
                            ou.getOperatorTemp().getName()
                            ;
                            if
                            (opTemp.getName().Equals(input
                            [2]))
                            {
                                input [2] = "inputD";
                            }
                        }
                        else
                        {
                            input [2] = "inputD";
                        }
                    }
                }
                else
                {
                    ou.add();
                }
                this.show = false;
            }

            /**
             * Updates data for input node E with name of connected
             * operator
            **/
            if (action == "connectE")
            {
                Operator opTemp = ou.getOperatorTemp();
                action = ou.clickedAt(newDown);
                string [] input = opTemp.getInput();
                if (action != "move")
                {
                    if (opTemp.getNumberOfInputNodes() == 2)
                    {
                        if (action == "output" || action == "body")
                        {
                            input [1] =
                            ou.getOperatorTemp().getName()
                            ;
```

```
                    if
                    (opTemp.getName().Equals(input
                    [1]))
                    {
                            input [1] = "inputE";
                    }
            }
            else
            {
                    input [1] = "inputE";
            }
    }
    if (opTemp.getNumberOfInputNodes() == 3)
    {
            if (action == "output" || action == "body")
            {
                    input          [2]          =
                    ou.getOperatorTemp().getName()
                    ;
                    if
                    (opTemp.getName().Equals(input
                    [2]))
                    {
                            input [2] = "inputE";
                    }
            }
            else
            {
                    input [2] = "inputE";
            }
    }
    if (opTemp.getNumberOfInputNodes() == 4)
    {
            if (action == "output" || action == "body")
            {
                    input          [3]          =
                    ou.getOperatorTemp().getName()
                    ;
                    if
                    (opTemp.getName().Equals(input
                    [3]))
                    {
                            input [3] = "inputE";
                    }
            }
            else
```

```
                        {
                                input [3] = "inputE";
                        }
                }
        }
        else
        {
                ou.add();
        }
        this.show = false;
}

/**
 * Updates data for input node C with name of connected
 * operator
**/
if (action == "connectC")
{
        Operator opTemp = ou.getOperatorTemp();
        action = ou.clickedAt(newDown);
        string [] input = opTemp.getInput();
        if (action != "move")
        {
                if (opTemp.getNumberOfInputNodes() == 3)
                {
                        if (action == "output" || action == "body")
                        {
                                input [1] =
                                ou.getOperatorTemp().getName()
                                ;
                                if
                                (opTemp.getName().Equals(input
                                [1]))
                                {
                                        input [1] = "inputC";
                                }
                        }
                        else
                        {
                                input [1] = "inputC";
                        }
                }
        }
        else
        {
                ou.add();
```

```
            }
            this.show = false;
        }
        else
        {
            action = ou.clickedAt(newDown);
            Operator temp = ou.getOperatorTemp();
            string name = temp.getName();

            /**Update action flag or action taken when body or
             * output node of operator clicked
            **/
            switch (action)
            {
                case "body":
                    if (this.show == true)
                    {
                        if (temp.getType()=="Table")
                        {
                            TableForm table = new
                            TableForm();
                            table.localDatabase
                            (xmfFilePath,temp.getQue
                            ry());
                            if (table.ShowDialog()==
                            DialogResult.Cancel)
                            {

                                table.Dispose();
                            }
                            else
                            {

                                table.Dispose();
                                string tableOutput =
                                table.getChosenTa
                                ble();
                                temp.setQuery
                                (tableOutput);
                                temp.setDbLoc
                                (table.getChosenDa
                                tabase());
                            }

                        }
                        if (temp.getType()=="Criteria")
```

```
                {
                    CriteriaForm criteria = new
                    CriteriaForm();
                    ArrayList criteriaList = new
                    ArrayList();
                    criteria.localDatabase(xmf
                    FilePath,temp.getQuery());
                    if      (criteria.ShowDialog()
                    ==DialogResult.Cancel)
                    {
                        criteria.Dispose();
                    }
                    else
                    {
                        criteria.Dispose();
                        string  criteriaOutput
                        =
                        criteria.getCriteria();
                        temp.setQuery
                        (criteriaOutput);
                        temp.setDbLoc
                        (dbaseLoc);
                    }
                }
            }
            else
            {
                this.show = true;
            }
            break;
        case "output":
            DataTable                rslt                =
            ou.processOutput(temp.getName());
            this.resultGrid.DataSource = rslt;
            this.tabControl.SelectedIndex = 3;
            break;
        case "inputA":
            action = "connectA";
            break;
        case "inputB":
            action = "connectB";
            break;
        case "inputC":
            action = "connectC";
            break;
        case "inputD":
```

```
                                action = "connectD";
                                break;
                        case "inputE":
                                action = "connectE";
                                break;
                        default:

                                break;
                }
        }
        this.panelB.Invalidate();
}

/**
 * Update location on mouse move
**/
private void DFQL_MouseMove(object sender, MouseEventArgs f)
{
        if(action == "move")
        {
                Point newDown = new Point(0,0);
                newDown.X = f.X;
                newDown.Y = f.Y;
                ou.move(newDown);
                this.panelB.Invalidate();
        }
}

/**
 * Add temporary operator back to list after move
**/
private void DFQL_MouseUp(object sender, MouseEventArgs g)
{
        if (action == "move")
        {
                ou.add();
                action = "none";
                this.panelB.Invalidate();
        }
}

/**
 * Reads operator list to paint each operator and
 * connections.
 * Each operator is painted differently depending
 * on type
```

```
**/
private void DFQL_Paint(object sender, PaintEventArgs e)
{
        int width = 55;
        int height = 25;
        Font opFont = new
        Font(FontFamily.GenericSansSerif,7,FontStyle.Regular);

        for (int i=0; i!= ou.count(); i++)
        {
                Operator opTemp = new Operator();
                opTemp = ou.getOperator(i);
                Point p = opTemp.getPoint();
                int x = p.X;
                int y = p.Y;
                string type = opTemp.getType();
                int alignX = 30-(type.Length*6/2);
                int inputCount = opTemp.getNumberOfInputNodes();
                string [] input = opTemp.getInput();

                e.Graphics.DrawRectangle(new
                Pen(Color.Black),x,y,width,height);
                e.Graphics.FillRectangle(new
                SolidBrush(Color.Blue),x,y,5,5);
                e.Graphics.FillRectangle(new
                SolidBrush(Color.Black),x+50,y,5,5);

                e.Graphics.FillEllipse(new
                SolidBrush(Color.DarkBlue),x+25,y+25,5,5);

                if ((type == "Table" || type == "Criteria") &&
                (opTemp.getQuery()!=null && opTemp.getQuery() !=
                ""))
                {
                        string opText = opTemp.getQuery();
                        Graphics g = e.Graphics;
                        SizeF size = g.MeasureString(opText,opFont);
                        g.Dispose();

                        /**Prints out only text that will fit operator box**/
                        if((int)size.Width > width)
                        {
                                int position = 0;
                                while(opText.IndexOfAny(new
                                char[]{'.',',',' ','=','!','<','>'},position)>0)
                                {
```

```
                int newPosition =
                opText.IndexOfAny(new
                char[]{'.',',',' ','=','!','<','>'},position);

                if(opText.IndexOf('.')==newPositio
                n)
                {
                        opText =
                        opText.Remove(position,n
                        ewPosition-position+1);
                        position = 0;
                }
                else
                {
                        position = newPosition + 1;
                }
        }
        Graphics f = e.Graphics;
        size = f.MeasureString(opText,opFont);
        f.Dispose();
        if ((int)size.Width > width)
        {
                Rectangle rc = new
                Rectangle(x+1,y+5,54,20);
                e.Graphics.DrawString(opText,op
                Font,new
                SolidBrush(Color.Black), rc);

                e.Graphics.DrawString("...",
                opFont,new
                SolidBrush(Color.Black),
                x+24,y+12);
        }
        else
        {
                int xAdjust = (x+(width/2))-
                (int)(size.Width/2);
                e.Graphics.DrawString(opText,op
                Font,new
                SolidBrush(Color.Black),
                xAdjust,y+7);
        }
}
else
{
```

```
                int xAdjust = (x+(width/2))-
                (int)(size.Width/2);
                e.Graphics.DrawString(opText,opFont,
                new
                SolidBrush(Color.Black),xAdjust,y+7);
        }
}
else
{
        Graphics h = e.Graphics;
        SizeF size = h.MeasureString(type,Font);
        h.Dispose();
        int xAdjust = (x+(width/2))-(int)(size.Width/2);
        e.Graphics.DrawString(type,Font,new
        SolidBrush(Color.Black), xAdjust,y+5);
}

if (inputCount == 2)
{
        if (input[0]!= "inputA")
        {
                Point pt = ou.location(input[0]);
                if (pt.X == 0 && pt.Y == 0)
                {
                        Operator op =
                        ou.getOperatorTemp();
                        pt = op.getPoint();
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+12,y-
                        2,pt.X+27,pt.Y+27);
                }
                else
                {
                        pt = ou.location(input[0]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+12,y-
                        2,pt.X+27,pt.Y+27);
                }
        }
        if (input[1]!= "inputE")
        {
                Point pt = ou.location(input[1]);
                if (pt.X == 0 && pt.Y == 0)
                {
                        Operator op =
                        ou.getOperatorTemp();
```

```
                    pt = op.getPoint();
                    e.Graphics.DrawLine(new
                    Pen(Color.Black),x+42,y-
                    2,pt.X+27,pt.Y+27);
                }
                else
                {
                    pt = ou.location(input[1]);
                    e.Graphics.DrawLine(new
                    Pen(Color.Black),x+42,y-
                    2,pt.X+27,pt.Y+27);
                }
            }
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Green), x+10,y-5,5,5);

            if (type == "Union")
            {
                e.Graphics.FillEllipse(new
                SolidBrush(Color.Green),x+40,y-5,5,5);
            }
            else
            {
                e.Graphics.FillEllipse(new
                SolidBrush(Color.Red),x+40,y-5,5,5);
            }
        }
        if (inputCount == 3)
        {
            if (type == "Join" || type == "EqJoin" || type ==
            "Diff"   || type == "Intersect")
            {
                e.Graphics.FillEllipse(new
                SolidBrush(Color.Green),x+10,y-5,5,5);
                e.Graphics.FillEllipse(new
                SolidBrush(Color.Green),x+25,y-5,5,5);
                e.Graphics.FillEllipse(new
                SolidBrush(Color.Red),x+40,y-5,5,5);
            }
            if (type == "GrpCnt" || type == "GrpAllSat" ||
            type == "GrpMin"    || type == "GrpMax" || type
            == "GrpAvg")
            {
                e.Graphics.FillEllipse(new
                SolidBrush(Color.Green),x+10,y-5,5,5);
```

```
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Red),x+25,y-5,5,5);
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Red),x+40,y-5,5,5);
        }
        if (input[0]!= "inputA")
        {
            Point pt = ou.location(input[0]);
            if (pt.X == 0 && pt.Y == 0)
            {
                Operator op =
                ou.getOperatorTemp();
                pt = op.getPoint();
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+12,y-
                2,pt.X+27,pt.Y+27);
            }
            else
            {
                pt = ou.location(input[0]);
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+12,y-
                2,pt.X+27,pt.Y+27);
            }
        }
        if (input[1]!= "inputC")
        {
            Point pt = ou.location(input[1]);
            if (pt.X == 0 && pt.Y == 0)
            {
                Operator op =
                ou.getOperatorTemp();
                pt = op.getPoint();
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+27,y-
                2,pt.X+27,pt.Y+27);
            }
            else
            {
                pt = ou.location(input[1]);
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+27,y-
                2,pt.X+27,pt.Y+27);
            }
        }
        if (input[2]!= "inputE")
```

```
        {
            Point pt = ou.location(input[2]);
            if (pt.X == 0 && pt.Y == 0)
            {
                Operator op =
                ou.getOperatorTemp();
                pt = op.getPoint();
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+42,y-
                2,pt.X+27,pt.Y+27);
            }
            else
            {
                pt = ou.location(input[2]);
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+42,y-
                2,pt.X+27,pt.Y+27);
            }
        }
    }
    if (inputCount == 4)
    {
        if (type == "GrpNSat")
        {
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Green),x+10,y-5,5,5);
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Red),x+20,y-5,5,5);
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Red),x+30,y-5,5,5);
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Red),x+40,y-5,5,5);
        }
        if (input[0]!= "inputA")
        {
            Point pt = ou.location(input[0]);
            if (pt.X == 0 && pt.Y == 0)
            {
                Operator op =
                ou.getOperatorTemp();
                pt = op.getPoint();
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+12,y-
                2,pt.X+27,pt.Y+27);
            }
            else
```

```
            {
                pt = ou.location(input[0]);
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+12,y-
                2,pt.X+27,pt.Y+27);
            }
        }
        if (input[1]!= "inputB")
        {
            Point pt = ou.location(input[1]);
            if (pt.X == 0 && pt.Y == 0)
            {
                Operator op =
                ou.getOperatorTemp();
                pt = op.getPoint();
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+22,y-
                2,pt.X+27,pt.Y+27);
            }
            else
            {
                pt = ou.location(input[1]);
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+22,y-
                2,pt.X+27,pt.Y+27);
            }
        }
        if (input[2]!= "inputD")
        {
            Point pt = ou.location(input[2]);
            if (pt.X == 0 && pt.Y == 0)
            {
                Operator op =
                ou.getOperatorTemp();
                pt = op.getPoint();
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+32,y-
                2,pt.X+27,pt.Y+27);
            }
            else
            {
                pt = ou.location(input[2]);
                e.Graphics.DrawLine(new
                Pen(Color.Black),x+32,y-
                2,pt.X+27,pt.Y+27);
            }
```

```
                }
                if (input[3]!= "inputE")
                {
                        Point pt = ou.location(input[3]);
                        if (pt.X == 0 && pt.Y == 0)
                        {
                                Operator op =
                                ou.getOperatorTemp();
                                pt = op.getPoint();
                                e.Graphics.DrawLine(new
                                Pen(Color.Black),x+42,y-
                                2,pt.X+27,pt.Y+27);
                        }
                        else
                        {
                                pt = ou.location(input[3]);
                                e.Graphics.DrawLine(new
                                Pen(Color.Black),x+42,y-
                                2,pt.X+27,pt.Y+27);
                        }
                }
        }

}

/**
 * Operator currently moving is taken from temporary
 * operator rather from list.  This code makes sure
 * moving operator is also captured during paint update.
**/
if (action == "move")
{
        Operator opTemp = new Operator();
        opTemp = ou.getOperatorTemp();
        Point p = opTemp.getPoint();
        int x = p.X;
        int y = p.Y;
        string type = opTemp.getType();
        int alignX = 30-(type.Length*6/2);
        int inputCount = opTemp.getNumberOfInputNodes();
        string [] input = opTemp.getInput();

        e.Graphics.DrawRectangle(new
        Pen(Color.Black),opTemp.getX(),opTemp.getY(),widt
        h,height);
```

```
e.Graphics.FillRectangle(new
SolidBrush(Color.Blue),opTemp.getX(),opTemp.getY(
),5,5);
e.Graphics.FillRectangle(new
SolidBrush(Color.Black),opTemp.getX()+50,opTemp.
getY(),5,5);
e.Graphics.FillEllipse(new
SolidBrush(Color.DarkBlue),opTemp.getX()+25,opTe
mp.getY()+25,5,5);

if ((type == "Table" || type == "Criteria") &&
opTemp.getQuery()!=null)
{
        string opText = opTemp.getQuery();
        Graphics g = e.Graphics;

        SizeF size = g.MeasureString(opText,opFont);
        g.Dispose();
        if((int)size.Width > width)
        {
                int position = 0;
                while(opText.IndexOfAny(new
                char[]{'.',',',' ','=','!','<','>'},position)>0)
                {
                        int newPosition =
                        opText.IndexOfAny(new
                        char[]{'.',',',' ','=','!','<','>'},position);

                        if(opText.IndexOf('.')==
                        newPosition)
                        {
                                opText =
                                opText.Remove(position,
                                newPosition-position+1);
                                position = 0;
                        }
                        else
                        {
                                position = newPosition + 1;
                        }
                }
                Graphics f = e.Graphics;

                size = f.MeasureString(opText,opFont);
                f.Dispose();
                if ((int)size.Width > width)
```

```
                {
                    Rectangle rc = new
                    Rectangle(x+1,y+5,54,20);
                    e.Graphics.DrawString(opText,op
                    Font,new
                    SolidBrush(Color.Black), rc);
                    e.Graphics.DrawString("...",
                    opFont,new
                    SolidBrush(Color.Black),
                    x+24,y+12);
                }
                else
                {
                    int xAdjust = (x+(width/2))-
                    (int)(size.Width/2);

                    e.Graphics.DrawString(opText,op
                    Font,new
                    SolidBrush(Color.Black),
                    xAdjust,y+7);
                }
            }
            else
            {
                int xAdjust = (x+(width/2))-
                (int)(size.Width/2);
                e.Graphics.DrawString(opText,opFont,
                new SolidBrush(Color.Black),
                xAdjust,y+7);
            }
        }
        else
        {
            Graphics h = e.Graphics;
            SizeF size = h.MeasureString(type,Font);
            h.Dispose();
            int xAdjust = (x+(width/2))-(int)(size.Width/2);
            e.Graphics.DrawString(type,Font,new
            SolidBrush(Color.Black), xAdjust,y+5);
        }

        if (inputCount == 2)
        {
            e.Graphics.FillEllipse(new
            SolidBrush(Color.Green),x+10,y-5,5,5);
            if (input[0]!= "inputA")
```

```
                {
                        Point pt = ou.location(input[0]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+12,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (input[1]!= "inputE")
                {
                        Point pt = ou.location(input[1]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+42,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (type == "Union")
                {
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Green),x+40,y-5,5,5);
                }
                else
                {
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+40,y-5,5,5);
                }
        }
        if (inputCount == 3)
        {
                if (type == "Join" || type == "EqJoin" || type ==
                "Diff" || type == "Intersect")
                {
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Green),x+10,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Green),x+25,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+40,y-5,5,5);
                }
                if (type == "GrpCnt" || type == "GrpAllSat" ||
                type == "GrpMin" || type == "GrpMax" || type ==
                "GrpAvg")
                {
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Green),x+10,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+25,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+40,y-5,5,5);
```

```
                }

                if (input[0]!= "inputA")
                {
                        Point pt = ou.location(input[0]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+12,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (input[1]!= "inputC")
                {
                        Point pt = ou.location(input[1]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+27,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (input[2]!= "inputE")
                {
                        Point pt = ou.location(input[2]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+42,y-
                        2,pt.X+27,pt.Y+27);
                }
        }
        if (inputCount == 4)
        {
                if (type == "GrpNSat")
                {
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Green),x+10,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+20,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+30,y-5,5,5);
                        e.Graphics.FillEllipse(new
                        SolidBrush(Color.Red),x+40,y-5,5,5);
                }
                if (input[0]!= "inputA")
                {
                        Point pt = ou.location(input[0]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+12,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (input[1]!= "inputB")
                {
```

```
                        Point pt = ou.location(input[1]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+22,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (input[2]!= "inputD")
                {
                        Point pt = ou.location(input[2]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+32,y-
                        2,pt.X+27,pt.Y+27);
                }
                if (input[3]!= "inputE")
                {
                        Point pt = ou.location(input[3]);
                        e.Graphics.DrawLine(new
                        Pen(Color.Black),x+42,y-
                        2,pt.X+27,pt.Y+27);
                }
        }
    }
}

/**Creates Select operator after clicking Select menu item and
 * sets action**/
private void selectMenuItem_Click(object sender,
System.EventArgs e)
{
    op = new Operator(2,"Select" + counter.ToString(),"Select");
    action = "create";
}

/**Creates Project operator after clicking Project menu item and
 * sets action**/
private void projectMenuItem_Click(object sender,
System.EventArgs e)
{
    op = new Operator(2,"Project" +
    counter.ToString(),"Project");
    action = "create";
}

/**Creates Join operator after clicking Join menu item and
 * sets action**/
private void joinMenuItem_Click(object sender, System.EventArgs
e)
```

```
{
        op = new Operator(3,"Join" + counter.ToString(),"Join");
        action = "create";
}

/**Creates Difference operator after clicking Diff menu item and
 * sets action**/
private void diffMenuItem_Click(object sender, System.EventArgs
e)
{
        op = new Operator(3,"Diff" + counter.ToString(),"Diff");
        action = "create";
}

/**Creates Union operator after clicking Union menu item and
 * sets action**/
private void unionMenuItem_Click(object sender,
System.EventArgs e)
{
        op = new Operator(2,"Union" + counter.ToString(),"Union");
        action = "create";
}

/**Creates Group Count operator after clicking GrpCnt menu item
 * and sets action**/
private void groupcntMenuItem_Click(object sender,
System.EventArgs e)
{
        op = new Operator(3,"GrpCnt" +
        counter.ToString(),"GrpCnt");
        action = "create";
}

/**Creates Table operator after clicking Table menu item and
 * sets action**/
private void tableMenuItem_Click(object sender, System.EventArgs
e)
{
        op = new Operator("Table" + counter.ToString(), "Table");
        action = "create";
}

/**Creates Criteria operator after clicking Criteria menu item and
 * sets action**/
private void criteriaMenuItem_Click(object sender,
System.EventArgs e)
```

```
{
        op = new Operator("Criteria" + counter.ToString(), "Criteria");
        action = "create";
}

/**Creates Eqjoin operator after clicking Eqjoin menu item and
 * sets action**/
private void eqjoinMenuItem_Click(object sender,
System.EventArgs e)
{
        op = new Operator(3,"EqJoin" + counter.ToString(),
        "EqJoin");
        action = "create";
}

/**Creates Group All Satisfy operator after clicking GrpAllSat menu
 * item and sets action**/
private void grpallsatMenuItem_Click(object sender,
System.EventArgs e)
{
        op = new Operator(3,"GrpAllSat" + counter.ToString(),
        "GrpAllSat");
        action = "create";
}

/**Creates Group N Satisfy operator after clicking GrpNSat menu
*item and sets action**/
private void groupnsatMenuItem_Click(object sender,
System.EventArgs e)
{
        op = new Operator(4,"GrpNSat" + counter.ToString(),
        "GrpNSat");
        action = "create";
}

/**Creates Intersect operator after clicking Intersect menu item and
 * sets action**/
private void intersectMenuItem_Click(object sender,
System.EventArgs e)
{
        op = new Operator(3,"Intersect" + counter.ToString(),
        "Intersect");
        action = "create";
}
```

```
/**Creates Group Minimum operator after clicking GrpMin menu item and sets action**/
private void groupminMenuItem_Click(object sender, System.EventArgs e)
{
        op = new Operator(3,"GrpMin" + counter.ToString(), "GrpMin");
        action = "create";
}

/**Creates Group Maximum operator after clicking GrpMax menu item and sets action**/
private void groupmaxMenuItem_Click(object sender, System.EventArgs e)
{
        op = new Operator(3,"GrpMax" + counter.ToString(), "GrpMax");
        action = "create";
}

/**Creates Group Average operator after clicking GrpAvg menu item and sets action**/
private void groupavgMenuItem_Click(object sender, System.EventArgs e)
{
        op = new Operator(3,"GrpAvg" + counter.ToString(), "GrpAvg");
        action = "create";
}

/**Loads DFQL query file**/
private void loadMenuItem_Click(object sender, System.EventArgs e)
{
        this.localFile.Filter = "DFQL Queries(*.udo)|*.udo|All Files(*.*)|*.*";
        if (localFile.ShowDialog() == DialogResult.OK)
        {
                OperatorUtility ouTemp = new OperatorUtility();
                string result = null;
                StreamReader readerStream = new StreamReader(localFile.FileName);

                /**
                 * Read file and create Operator object and add to a
                 * temporary OperatorUtility
```

```
**/
while((result = readerStream.ReadLine())!=null)
{
        string [ ] data = result.Split('|');
        string name = data[0];
        string type = data[1];
        Point pt = new
        Point(Int32.Parse(data[2]),Int32.Parse
        (data[3]));
        int n = Int32.Parse(data[4]);
        Operator opTemp = new
        Operator(pt,n,name,type);

        if(n > 0)
        {
                string [] inputs = new string[n];
                for(int i = 0; i < n; i++)
                {
                        inputs[i]=data[i+5];
                }
                opTemp.setInput(inputs);
        }
        ouTemp.add(opTemp);
}
readerStream.Close();
OperatorUtility ouTempNew = new OperatorUtility();
int ouTempCount = ouTemp.count();
/**
 * Rename operator and update other operators
 * connected to it
**/
for(int u = 0; u < ouTemp.count(); u++)
{
        Operator opTemp = ouTemp.getOperator(u);
        string name = opTemp.getName() +
        this.counter.ToString();
        string type = opTemp.getType();
        int x = opTemp.getPoint().X;
        int y = opTemp.getPoint().Y;
        Point pt = new Point(x,y);
        int n = opTemp.getNumberOfInputNodes();

        for(int c = 0; c < ouTemp.count(); c++)
        {
                Operator find = ouTemp.getOperator(c);
                string nameTest = find.getName();
```

```
                                int m = find.getNumberOfInputNodes();
                                if ( m > 0)
                                {
                                        string [ ] ins = find.getInput();
                                        for(int g = 0; g < m; g++)
                                        {
                                                if(ins[g]                       ==
                                                opTemp.getName())
                                                {
                                                        ins[g] = name;
                                                }
                                        }
                                        ouTemp.getOperator(c).setInput
                                        (ins);
                                }
                        }
                        if ( n > 0 )
                        {
                                string [] inputs = opTemp.getInput();
                                opTemp = new Operator(pt, n, name,
                                type);
                                opTemp.setInput(inputs);
                        }
                        else
                        {
                                opTemp = new Operator(pt, n, name,
                                type);
                        }
                        ouTemp.replace(opTemp,u);
                        counter++;
                }

                /**
                 * Add renamed and updated user defined operators
                **/
                for(int v = 0; v < ouTempCount; v++)
                {
                        ou.add(ouTemp.getOperator(v));
                }
        }
        this.localFile.Filter = "SQLCE Databases(*.sdf)|*.sdf|All
        Files(*.*)|*.*";
}

/**Saves current DFQL query to file**/
```

```
private void saveMenuItem_Click(object sender, System.EventArgs
e)
{
        if(saveFile.ShowDialog()==DialogResult.OK)
        {
                StreamWriter saveStream = new
                StreamWriter(saveFile.FileName, false);
                Operator opTemp = new Operator();

                for(int o = 0; o < ou.count(); o++)
                {
                        opTemp = ou.getOperator(o);
                        string data = null;
                        data = data + opTemp.getName() + "|";
                        data = data + opTemp.getType() + "|";
                        data = data + opTemp.getX() + "|";
                        data = data + opTemp.getY() + "|";
                        data = data +
                        opTemp.getNumberOfInputNodes() + "|";
                        for (int t = 0; t <
                        opTemp.getNumberOfInputNodes(); t++)
                        {
                                data = data + opTemp.getInput()[t] + "|";
                        }
                        saveStream.WriteLine(data);
                }
                saveStream.Close();
                ou = new OperatorUtility();
        }
}


private void RDAPull()
{
        if (! System.IO.File.Exists(@"\My
        Documents\Northwind.sdf"))
        {
                SqlCeEngine eng = new SqlCeEngine(@"Data
                Source=\My Documents\Northwind.sdf");
                eng.CreateDatabase();
                eng.Dispose();
        }
        SqlCeConnection cn = new SqlCeConnection(@"Data
        Source=\My Documents\Northwind.sdf");
        SqlCeCommand cmd = new SqlCeCommand("DROP
        TABLE Customers", cn);
```

```
SqlCeCommand cmd2 = new SqlCeCommand("DROP
TABLE Employees", cn);
SqlCeCommand cmd3 = new SqlCeCommand("DROP
TABLE Categories", cn);
SqlCeCommand cmd4 = new SqlCeCommand("DROP
TABLE CustomerCustomerDemo", cn);
SqlCeCommand cmd5 = new SqlCeCommand("DROP
TABLE CustomerDemographics", cn);
SqlCeCommand cmd6 = new SqlCeCommand("DROP
TABLE EmployeeTerritories", cn);
SqlCeCommand cmd7 = new SqlCeCommand("DROP
TABLE Order_Details", cn);
SqlCeCommand cmd8 = new SqlCeCommand("DROP
TABLE Orders", cn);
SqlCeCommand cmd9 = new SqlCeCommand("DROP
TABLE Products", cn);
SqlCeCommand cmd10 = new SqlCeCommand("DROP
TABLE Region", cn);
SqlCeCommand cmd11 = new SqlCeCommand("DROP
TABLE Shippers", cn);
SqlCeCommand cmd12 = new SqlCeCommand("DROP
TABLE Supplies", cn);
SqlCeCommand cmd13 = new SqlCeCommand("DROP
TABLE Territories", cn);
cn.Open();
try
{
        cmd.ExecuteNonQuery();
        cmd2.ExecuteNonQuery();
        cmd3.ExecuteNonQuery();
        cmd4.ExecuteNonQuery();
        cmd5.ExecuteNonQuery();
        cmd6.ExecuteNonQuery();
        cmd7.ExecuteNonQuery();
        cmd8.ExecuteNonQuery();
        cmd9.ExecuteNonQuery();
        cmd10.ExecuteNonQuery();
        cmd11.ExecuteNonQuery();
        cmd12.ExecuteNonQuery();
        cmd13.ExecuteNonQuery();
}
catch (SqlCeException sqlCeEx)
{

}
cn.Close();
```

```
cn.Dispose();
SqlCeRemoteDataAccess rda = new SqlCeRemoteDataAccess();
string sCon = @"Provider=SQLOLEDB;Data Source=MyComputer;" + @"Initial Catalog=Northwind;integrated security=SSPI;"
+ @"Persist Security Info=False"; rda.InternetUrl = @"http://MyComputer/Northwind/sscesa20.dll";
rda.LocalConnectionString = @"Data Source=\My Documents\Northwind.sdf";
try
{
        rda.Pull("Customers", "SELECT * FROM Customers",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Employees", "SELECT * FROM Employees",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Categories", "SELECT * FROM Categories",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("CustomerCustomerDemo", "SELECT * FROM CustomerCustomerDemo",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("CustomerDemographics", "SELECT * FROM CustomerDemographics",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("EmployeeTerritories", "SELECT * FROM EmployeeTerritories",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Order_Details", "SELECT * FROM [Order Details]",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Orders", "SELECT * FROM Orders",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Products", "SELECT * FROM Products",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Region", "SELECT * FROM Region",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Shippers", "SELECT * FROM Shippers",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Suppliers", "SELECT * FROM Suppliers",
                sCon, RdaTrackOption.TrackingOff);
        rda.Pull("Territories", "SELECT * FROM Territories",
                sCon, RdaTrackOption.TrackingOff);
}
catch (SqlCeException sqlCeEx)
{
}
```

```
        rda.Dispose();
}

private void menuRemoteNorthwind_Click(object sender,
System.EventArgs e)
{
        RDAPull();
        string file = @"\My Documents\Northwind.sdf";
        createTreeView(file);
}

private void createTreeView(string fileName)
{
        DatabaseUtility duDbNames = new DatabaseUtility();
        string temp = duDbNames.getDatabaseName(fileName);
        this.xmlDoc = new XmlDocument();
        XmlTextReader file = new XmlTextReader(xmfFilePath);
        this.xmlDoc.Load(file);
        file.Close();
        XmlNode firstChildNode = xmlDoc.DocumentElement;

        XmlElement newElement = xmlDoc.CreateElement(temp);
        firstChildNode.AppendChild(newElement);
        TreeNode node = new TreeNode(newElement.Name);
        node.ImageIndex = 1;
        node.SelectedImageIndex = 1;
        rootNode.Nodes.Add(node);
        ArrayList tables = new ArrayList();
        DatabaseUtility duNames = new DatabaseUtility();
        tables = duNames.getTableNames(fileName);
        for(int t=0; t<tables.Count;t++)
        {
                XmlElement tableElement =
                xmlDoc.CreateElement((string)tables[t]);
                newElement.AppendChild(tableElement);

                TreeNode tblNode = new
                TreeNode(tableElement.Name);
                tblNode.ImageIndex = 2;
                tblNode.SelectedImageIndex = 2;
                node.Nodes.Add(tblNode);
                ArrayList columns = new ArrayList();
                columns = duNames.getColumnNames
                (fileName,tableElement.Name);
                for(int c=0; c<columns.Count;c++)
                {
```

```
                                XmlElement columnElement =
                                xmlDoc.CreateElement ((string)columns[c]);
                                tableElement.AppendChild(columnElement);
                                TreeNode colNode = new
                                TreeNode(columnElement.Name);
                                colNode.ImageIndex = 3;
                                colNode.SelectedImageIndex = 3;

                                tblNode.Nodes.Add(colNode);

                        }
                }
                treeView.ExpandAll();
                dbaseLoc = fileName;
                xmlDoc.Save(xmfFilePath);
        }

        private void hScrollBar_ValueChanged(object sender,
        System.EventArgs e)
        {
                panelB.Left = -1 * hScrollBar.Value;
        }

        private void vScrollBar_ValueChanged(object sender,
        System.EventArgs e)
        {
                panelB.Top = -1 * vScrollBar.Value;
        }


    }
}
```

2. Operator.cs

```
using System;
using System.Drawing;

namespace DFQL
{
	/**
	 * Operator class for storage of information regarding name,
	 * location, type, number of inputs, inputs, database, and query
	**/
	public class Operator
	{
		private int x;
		private int y;
		private int numberOfInputNodes;
		private string name;
		private string [] input;
		private string type;
		private string query;
		private string dbLocation;


		/**
		 * Operator constructor
		 * Initializes operator when no parameters received
		**/
		public Operator()
		{
			this.setX(0);
			this.setY(0);
			this.setNoOfInputNodes(0);
			this.setName(null);
			this.setType(null);
			this.setQuery(null);

		}

		/**
		 * Operator constructor
		 * Initializes operator when only name and type received
		 * as parameters
		**/
		public Operator(string opName, string opType)
		{
			this.setX(0);
			this.setY(0);
```

```
            this.setNoOfInputNodes(0);
            this.setName(opName);
            this.setType(opType);
            this.setQuery(null);

        }

        /**
         * Operator constructor
         * Initializes operator when only number of input nodes,
         * name, and type received as parameters
        **/
        public Operator(int nuInputNode,string opName, string opType)
        {
            this.setX(0);
            this.setY(0);
            this.setNoOfInputNodes(nuInputNode);
            this.setName(opName);
            this.setType(opType);
            this.setQuery(null);
            if(nuInputNode == 2)
            {
                input = new string [2];
                this.input[0] = "inputA";
                this.input[1] = "inputE";
                switch (opType)
                {
                    case "Select":

                        break;
                    default:
                        break;
                }
            }
            if(nuInputNode == 3)
            {
                input = new string [3];
                this.input[0] = "inputA";
                this.input[1] = "inputC";
                this.input[2] = "inputE";
}
            if(nuInputNode == 4)
            {
                input = new string [4];
                this.input[0] = "inputA";
                this.input[1] = "inputB";
```

```
                this.input[2] = "inputD";
                this.input[3] = "inputE";
        }

}

/**
 * Operator constructor
 * Initializes operator when point location, number
 * of input nodes, name, and type are received as
 * parameters
**/
public Operator(Point p, int nuInputNode, string opName, string
opType)
{
        this.setX(p.X);
        this.setY(p.Y);
        this.setNoOfInputNodes(nuInputNode);
        this.setName(opName);
        this.setType(opType);
        this.setQuery(null);
        this.input = new string[nuInputNode];
        if(nuInputNode == 2)
        {
                this.input[0] = "inputA";
                this.input[1] = "inputE";
        }
        if(nuInputNode == 3)
        {
                this.input[0] = "inputA";
                this.input[1] = "inputC";
                this.input[2] = "inputE";
        }
        if(nuInputNode == 4)
        {
                this.input[0] = "inputA";
                this.input[1] = "inputB";
                this.input[2] = "inputD";
                this.input[3] = "inputE";
        }

}

/**Sets the number of input nodes**/
private void setNoOfInputNodes (int inputNodeCount)
{
```

```
        this.numberOfInputNodes = inputNodeCount;
}

/**Sets the name of operator**/
private void setName (string operatorName)
{
        this.name = operatorName;
}

/**Sets the type of operator**/
private void setType (string operatorType)
{
        this.type = operatorType;
}

/**Sets the x-coordinate of operator**/
public void setX (int xCoordinate)
{
        this.x = xCoordinate;
}

/**Sets the y-coordinate of operator**/
public void setY (int yCoordinate)
{
        this.y = yCoordinate;
}

/**Sets the inputs of operator**/
public void setInput (string [] inputStrings)
{
        inputStrings.CopyTo(this.input,0);
}

/**Sets the query of operator**/
public void setQuery (string operatorQuery)
{
        this.query = operatorQuery;
}

/**Sets database location of operator**/
public void setDbLoc (string dbLoc)
{
        this.dbLocation = dbLoc;
}

/**Gets the x-coordinate of operator**/
```

```
public int getX()
{
	return this.x;
}

/**Gets the y-coordinate of opertor**/
public int getY()
{
	return this.y;
}

/**Gets the point of operator using x and y coordinates**/
public Point getPoint()
{
	Point pt = new Point(this.x, this.y);
	return pt;
}

/**Gets number of input nodes of operator**/
public int getNumberOfInputNodes()
{
	return this.numberOfInputNodes;
}

/**Gets name of operator**/
public string getName()
{
	return this.name;
}

/**Gets inputs of operator**/
public string [] getInput()
{
	return this.input;
}

/**Gets type of operator**/
public string getType()
{
	return this.type;
}

/**Gets query of operator**/
public string getQuery()
{
	return this.query;
```

```
                }

                /**Gets database location of operator**/
                public string getDbLoc()
                {
                        return this.dbLocation;
                }
        }
}
```

3. AddLocalDatabase.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlServerCe;

namespace DFQL
{
        /**
         * Creates a local database called Enrollment.  It has two purposes.
         * One is to have a local database to create DFQL queries againts.
         * Second is for creating temporary tables to support DFQL queries.
        **/
        public class AddLocalDatabase
        {
                private string strFile = @"My Documents\Enrollment.sdf";
                private string strConn = "Data Source=" + @"My
                Documents\Enrollment.sdf";

                /**
                 * AddLocalDatabase constructor
                 * Delete database if it exists else create one using SQL CE
                **/
                public AddLocalDatabase()
                {
                        if (File.Exists(strFile))
                        {
                                File.Delete(strFile);
                        }
                        SqlCeEngine dbEngine = new SqlCeEngine();
                        dbEngine.LocalConnectionString = strConn;
                        try
                        {
                                dbEngine.CreateDatabase();
                        }
                        catch ( SqlCeException exSQL )
                        {
                                MessageBox.Show("Unable to create database at " +
                                        strFile +
                                        ". Reason: " +
                                        exSQL.Errors[0].Message);
                        }
                        populateDB();
                        dbEngine.Dispose();
                }
```

```
/**
 * Creates Tables and Columns and inserts predefined data
**/
private void populateDB()
{
        SqlCeConnection connDB = new SqlCeConnection();
        SqlCeCommand cmdDB = new SqlCeCommand();

        connDB.ConnectionString = strConn;
        connDB.Open();

        cmdDB.Connection = connDB;

        cmdDB.CommandText =
                " CREATE TABLE Course " +
                " (CNO nchar(5) not null " +
                " , TITLE nchar(25) not null " +
                " , INO nchar(5) not null " +
                " )";

        cmdDB.ExecuteNonQuery();


        cmdDB.CommandText =
                " INSERT Course " +
                "(CNO, TITLE, INO) " +
                "VALUES ('C001','Course01', 'I001')";

        cmdDB.ExecuteNonQuery();

        cmdDB.CommandText =
                " INSERT Course " +
                "(CNO, TITLE, INO) " +
                "VALUES ('C002','Course02', 'I002')";

        cmdDB.ExecuteNonQuery();

        cmdDB.CommandText =
                " INSERT Course " +
                "(CNO, TITLE, INO) " +
                "VALUES ('C003','Course03', 'I001')";

        cmdDB.ExecuteNonQuery();

        cmdDB.CommandText =
```

```
        " INSERT Course " +
        "(CNO, TITLE, INO) " +
        "VALUES ('C004','Course04', 'I002')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " CREATE TABLE Instructor " +
        " (INO nchar(5) not null " +
        " , INAME nchar(25) not null " +
        " )";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Instructor " +
        "(INO, INAME) " +
        "VALUES ('I001','Instructor01')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Instructor " +
        "(INO, INAME) " +
        "VALUES ('I002','Instructor02')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Instructor " +
        "(INO, INAME) " +
        "VALUES ('I003','Instructor03')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " CREATE TABLE Enroll " +
        " (ECno nchar(5) not null " +
        " , ESno nchar(12) not null " +
        " , GRADE nchar(2) not null " +
        " )";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
```

```
        "(ECno, ESno, GRADE) " +
        "VALUES ('C001','111-11-1111','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C001','111-11-1112','B')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C001','111-11-1113','B')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C001','111-11-1114','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C001','111-11-1115','B')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C001','111-11-1116','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C002','111-11-1111','A')";

cmdDB.ExecuteNonQuery();
```

```
cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C002','111-11-1112','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C003','111-11-1113','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C003','111-11-1114','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C003','111-11-1115','B')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C003','111-11-1116','B')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Enroll " +
        "(ECno, ESno, GRADE) " +
        "VALUES ('C004','111-11-1112','A')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " CREATE TABLE Student " +
        " (SNO nchar(11) not null " +
```

```
        " , SNAME nchar(25) not null " +
        " , AGE int " +
        " , GENDER nchar(6) not null " +
        " )";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
" INSERT Student (SNO, SNAME, AGE, GENDER) VALUES
('111-11-1111','Student01',22,'female')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Student " +
        "(SNO, SNAME, AGE, GENDER) " +
        "VALUES ('111-11-1112','Student02'" +
        ",25,'male')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Student " +
        "(SNO, SNAME, AGE, GENDER) " +
        "VALUES ('111-11-1113','Student03'" +
        ",31,'female')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Student " +
        "(SNO, SNAME, AGE, GENDER) " +
        "VALUES ('111-11-1114','Student04'" +
        ",38,'male')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
        " INSERT Student " +
        "(SNO, SNAME, AGE, GENDER) " +
        "VALUES ('111-11-1115','Student05'" +
        ",27,'female')";

cmdDB.ExecuteNonQuery();

cmdDB.CommandText =
```

```
                    " INSERT Student " +
                    "(SNO, SNAME, AGE, GENDER) " +
                    "VALUES ('111-11-1116','Student06'" +
                    ",29,'male')";

                cmdDB.ExecuteNonQuery();
                cmdDB.Dispose();
                connDB.Close();
            }
        }
}
```

4. Table.cs

```
using System;
using System.Xml;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace DFQL
{
	/**
	 * Class to choose table from a tree.
	**/
	public class TableForm : System.Windows.Forms.Form
	{
		private System.Windows.Forms.ImageList imageList;
		private System.Windows.Forms.TreeView treeView;
		private System.Windows.Forms.Button okButton;
		private System.Windows.Forms.TreeNode rootNode;
		private string chosenTable = null;
		private string chosenDatabase = null;

		/**
		 * TableForm constructor
		**/
		public TableForm()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			//TODO: Add any constructor code after InitializeComponent
			call
			//
			this.rootNode = new System.Windows.Forms.TreeNode();
			this.rootNode.Text = "Database";
			this.treeView.Nodes.Add(rootNode);
			treeView.AfterSelect += new
			TreeViewEventHandler(nodeChosen);

		}

		/// <summary>
		/// Clean up any resources being used.
```

```
/// </summary>
protected override void Dispose( bool disposing )
{
	base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
	System.Resources.ResourceManager resources = new
	System.Resources.ResourceManager(typeof(TableForm));
	this.okButton = new System.Windows.Forms.Button();
	this.treeView = new System.Windows.Forms.TreeView();
	this.imageList = new System.Windows.Forms.ImageList();

	//
	// okButton
	//
	this.okButton.Location = new System.Drawing.Point(88,
	240);
	this.okButton.Size = new System.Drawing.Size(64, 24);
	this.okButton.Text = "OK";
	this.okButton.Click += new
	System.EventHandler(this.okButton_Click);

	//
	// treeView
	//
	this.treeView.ImageList = this.imageList;
	this.treeView.Location = new System.Drawing.Point(8, 8);
	this.treeView.Size = new System.Drawing.Size(224, 224);

	//
	// imageList
	//
	this.imageList.Images.Add(((System.Drawing.Image)
	(resources.GetObject("resource"))));
	this.imageList.Images.Add(((System.Drawing.Image)
	(resources.GetObject("resource1"))));
	this.imageList.Images.Add(((System.Drawing.Image)
	(resources.GetObject("resource2"))));
```

```
            this.imageList.ImageSize = new System.Drawing.Size(16,
            16);

            //
            // TableForm
            //
            this.Controls.Add(this.treeView);
            this.Controls.Add(this.okButton);
            this.Text = "TableForm";

    }
    #endregion

    /**
     * OK button after choosing table
    **/
    private void okButton_Click(object sender, System.EventArgs e)
    {
            this.DialogResult = DialogResult.OK;
    }

    /**
     * Loads database schema into a tree from XML document
    **/
    public void localDatabase(string xmlFileLoc, string output)
    {

            XmlDocument dom = new XmlDocument();
            dom.Load(xmlFileLoc);
            this.treeView.Nodes.Clear();
            this.treeView.Nodes.Add(new
            TreeNode(dom.DocumentElement.Name));
            TreeNode tNode = new TreeNode();
            tNode = this.treeView.Nodes[0];
            AddNode(dom.DocumentElement,tNode,0);
            treeView.ExpandAll();
            chosenTable = output;
    }

    /**
     * Node chosen and determines which database chosen node
    belongs to.
    **/
    private void nodeChosen(System.Object sender,
            System.Windows.Forms.TreeViewEventArgs e)
    {
```

```
        if(e.Node.GetNodeCount(false)==0)
        {
            if(e.Node.Text != "Databases")
            {
                chosenTable = e.Node.Text;
            }
            int tableLoc = -1;
            try
            {
                tableLoc = e.Node.FullPath.IndexOf('\\',10);
            }
            catch
            {
            }
            if (tableLoc > -1)
            {
                chosenDatabase = @"\My Documents\" +
                e.Node.FullPath.Substring(10,tableLoc - 10) +
                ".sdf";
            }
        }
}

/**Get chosen table method**/
public string getChosenTable()
{
        return chosenTable;
}

/**Get chosen database method**/
public string getChosenDatabase()
{
        return chosenDatabase;
}

/**AddNode method for populating treenode with xmlnode**/
private void AddNode(XmlNode inXmlNode, TreeNode inTreeNode,
int counter)
{
        XmlNode xNode;
        TreeNode tNode;
        XmlNodeList nodeList;
        int index = counter;
        index++;
        if (inXmlNode.HasChildNodes)
        {
```

```
                    nodeList = inXmlNode.ChildNodes;
                    if(counter<2)
                    {
                        for (int i = 0; i<=nodeList.Count-1; i++)
                        {
                            xNode = inXmlNode.ChildNodes[i];
                            inTreeNode.ImageIndex = counter;
                            inTreeNode.SelectedImageIndex =
                            counter;
                            inTreeNode.Nodes.Add(new
                            TreeNode(xNode.Name));

                            tNode = inTreeNode.Nodes[i];
                            AddNode(xNode,tNode,index);

                        }
                    }
                    else
                    {
                        inTreeNode.ImageIndex = counter;
                        inTreeNode.SelectedImageIndex = counter;
                    }
                }
            }
        }
    }
```

5. Criteria.cs

```
using System;
using System.Xml;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace DFQL
{
	/**
	 * Class for creating criteria information easier for user.
	 * Displays a tree view of databases connected. User clicks
	 * on column nodes and is displayed in a text box.
	**/
	public class CriteriaForm : System.Windows.Forms.Form
	{
		private System.Windows.Forms.Button okButton;
		private System.Windows.Forms.TreeView treeView;
		private System.Windows.Forms.TextBox textBox;
		private System.Windows.Forms.ToolBar toolBar;
		private System.Windows.Forms.ImageList imageList;
		private System.Windows.Forms.TreeNode rootNode;
		private string criteria = null;

		/**
		 * CriteriaForm constructor
		**/
		public CriteriaForm()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent
			call
			//
			this.rootNode = new System.Windows.Forms.TreeNode();
			this.rootNode.Text = "Database";
			this.treeView.Nodes.Add(rootNode);
			treeView.AfterSelect += new
			TreeViewEventHandler(nodeChosen);
		}
```

```
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
	criteria = this.textBox.Text;
	base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
	System.Resources.ResourceManager resources = new
	System.Resources.ResourceManager(typeof(CriteriaForm));
	this.treeView = new System.Windows.Forms.TreeView();
	this.textBox = new System.Windows.Forms.TextBox();
	this.okButton = new System.Windows.Forms.Button();
	this.toolBar = new System.Windows.Forms.ToolBar();
	this.imageList = new System.Windows.Forms.ImageList();

	//
	// treeView
	//
	this.treeView.ImageList = this.imageList;
	this.treeView.Size = new System.Drawing.Size(240, 144);

	//
	// textBox
	//
	this.textBox.Location = new System.Drawing.Point(0, 152);
	this.textBox.Multiline = true;
	this.textBox.ScrollBars =
	System.Windows.Forms.ScrollBars.Vertical;
	this.textBox.Size = new System.Drawing.Size(240, 40);
	this.textBox.Text = "";

	//
	// okButton
	//
	this.okButton.Location = new System.Drawing.Point(88,
	200);
	this.okButton.Size = new System.Drawing.Size(56, 24);
```

```
            this.okButton.Text = "OK";
            this.okButton.Click += new
            System.EventHandler(this.okButton_Click);

            //
            // imageList
            //
            this.imageList.Images.Add(((System.Drawing.Image)
            (resources.GetObject("resource"))));
            this.imageList.Images.Add(((System.Drawing.Image)
            (resources.GetObject("resource1"))));
            this.imageList.Images.Add(((System.Drawing.Image)
            (resources.GetObject("resource2"))));
            this.imageList.Images.Add(((System.Drawing.Image)
            (resources.GetObject("resource3"))));
            this.imageList.ImageSize = new System.Drawing.Size(16,
            16);

            //
            // CriteriaForm
            //
            this.Controls.Add(this.okButton);
            this.Controls.Add(this.textBox);
            this.Controls.Add(this.treeView);
            this.Controls.Add(this.toolBar);
            this.Text = "CriteriaForm";

}
#endregion

/**
 * Loads database schema into a tree from XML document
**/
public void localDatabase(string xmlFileLoc, string output)
{
            XmlDocument dom = new XmlDocument();
            dom.Load(xmlFileLoc);
            this.treeView.Nodes.Clear();
            this.treeView.Nodes.Add(new
            TreeNode(dom.DocumentElement.Name));
            TreeNode tNode = new TreeNode();
            tNode = this.treeView.Nodes[0];
            AddNode(dom.DocumentElement,tNode,0);
            this.textBox.Text = output;
            treeView.ExpandAll();
}
```

```
/**
 * Determines which node is chosen and displays it
 * in text box
**/
private void nodeChosen(System.Object sender,
        System.Windows.Forms.TreeViewEventArgs e)
{
        if(e.Node.GetNodeCount(false)==0)
        {
                this.textBox.Text = this.textBox.Text +
                        e.Node.Parent.Text+"."+e.Node.Text;
        }
        else
        {
                if(e.Node.Text != "Databases")
                {
                        if (e.Node.Parent.Text != "Databases")
                        {
                                this.textBox.Text  =  this.textBox.Text  +
                                e.Node.Text;
                        }
                }
        }
}

/**Gets criteria in order not to re-enter it after closing form**/
public string getCriteria()
{
        return criteria;
}

/**Send OK when okButton clicked**/
private void okButton_Click(object sender, System.EventArgs e)
{
        this.DialogResult = DialogResult.OK;
}

/**AddNode method for populating treenode with xmlnode**/
private void AddNode(XmlNode inXmlNode, TreeNode inTreeNode,
int counter)
{
        XmlNode xNode;
        TreeNode tNode;
        XmlNodeList nodeList;
        int index = counter;
```

```
                index++;
                if (inXmlNode.HasChildNodes)
                {
                        nodeList = inXmlNode.ChildNodes;
                        for (int i = 0; i<=nodeList.Count-1; i++)
                        {
                                xNode = inXmlNode.ChildNodes[i];
                                inTreeNode.ImageIndex = counter;
                                inTreeNode.SelectedImageIndex = counter;
                                inTreeNode.Nodes.Add(new
                                TreeNode(xNode.Name));

                                tNode = inTreeNode.Nodes[i];
                                AddNode(xNode,tNode,index);

                        }
                }
                else
                {
                        inTreeNode.ImageIndex = counter;
                        inTreeNode.SelectedImageIndex = counter;
                }
        }

    }
}
```

6. OperatorUtility.cs

```
using System;
using System.Data;
using System.Collections;
using System.Drawing;
using System.Data.SqlServerCe;

namespace DFQL
{
	/**
	 * OperatorUtility class that keeps track of operators,
	 * determines which part of the operator was clicked on,
	 * operator location, appropriate query, and creation
	 * and deletion of temporary tables that act as view
	**/
	public class OperatorUtility
	{
		/**Arraylist for storing operators**/
		private System.Collections.ArrayList operatorList;

		/**Multi-purpose temporary operator**/
		private Operator op;

		/**
		 * OperatorUtility constructor
		 * Initializes data members
		**/
		public OperatorUtility()
		{
			this.operatorList = new ArrayList();
			this.op = new Operator();
		}

		/**Adds temporary operator to the list**/
		public void add ()
		{
			this.operatorList.Add(op);
		}

		/**Adds a given operator to the list**/
		public void add (Operator o)
		{
			this.operatorList.Add(o);
		}

		/**Counts the number of operators on the list**/
```

```
public int count()
{
        return this.operatorList.Count;
}

/**Removes an operator from the list given an index number**/
private void remove (int index)
{
        this.operatorList.RemoveAt(index);
}

/**
 * Replaces Operator at a given index with a given
 * Operator
**/
public void replace (Operator op, int index)
{
        operatorList[index] = op;
}
/**
 * Given a point, updates the x and y coordinates of
 * temporary operator
**/
public void move (Point p)
{
        op.setX(p.X);
        op.setY(p.Y);
}

/**Gets operator located at given index number**/
public Operator getOperator (int index)
{
        Operator opTemp = (Operator) operatorList[index];
        return opTemp;
}

/**Gets operator given a name**/
public Operator getOperator (string name)
{
        Operator opTemp = new Operator();
        Point p = new Point(0,0);
        p = location(name);
        int loc = location(p);
        if (loc > -1)
        {
                opTemp = (Operator) operatorList[loc];
```

```
        }
        return opTemp;
}

/**Gets temporary operator**/
public Operator getOperatorTemp()
{
        return op;
}

/**Determines point location of operator given a name**/
public Point location (string name)
{
        Point p = new Point(0,0);
        Operator opTemp = new Operator();
        for (int c=0;c!=operatorList.Count;c++)
        {
                opTemp = (Operator) operatorList[c];
                if (opTemp.getName() == name)
                {
                        p = opTemp.getPoint();
                        break;
                }
        }
        return p;
}

/**
 * Determines index location of operator given
 * a point location
**/
private int location (Point p)
{
        int result = -1;
        Point opPoint = new Point(0,0);
        Operator opTemp = new Operator();
        for (int d=0;d!=operatorList.Count;d++)
        {
                opTemp = (Operator) operatorList[d];
                opPoint = opTemp.getPoint();
                if (opPoint.X<=p.X && opPoint.X+55>=p.X)
                {
                        if (opPoint.Y-5<=p.Y && opPoint.Y+30>=p.Y)
                        {
                                result = d;
                                break;
```

```
                    }
                }
            }
            return result;
}

/**
 * Processes operator to determine output query and
 * returns result as a data table given the name of
 * the operator
**/
public DataTable processOutput(string name)
{
            Operator tempO = getOperator(name);
            string defaultDbLoc = @"\My Documents\Enrollment.sdf";
            string [] input = tempO.getInput();
            DataTable tempDt = new DataTable();
            for(int i=0;i<=tempO.getNumberOfInputNodes()-1;i++)
            {
                        /**Determines if operator connected to a Table**/
                        if(!input[i].StartsWith("Table"))
                        {
                                    /**Determines if operator connected to a
                                    Criteria**/
                                    if (!input[i].StartsWith("Criteria"))
                                    {
                                                /**Recursion to process all connected
                                                operators**/
                                                tempDt = processOutput(input[i]);
                                                if(i<1)
                                                {
                                                            /**create tableA**/
                                                            string query = "create table
                                                            TableA (";

                                                            /**Creates appropriate query
                                                            depending on type**/
                                                            foreach(DataColumn colName in
                                                            tempDt.Columns)
                                                            {
                                                                        query = query +
                                                                        colName.ColumnName;

                                                                        /**Determines column
                                                                        types**/
```

```
                string type = colName.DataType.ToString();


                switch(type)
                {
                    case "System.String":
                        query = query + " nchar(50),";
                        break;
                    case "System.Int32":
                        query = query + " int,";
                        break;
                    case "System.DateTime":
                        query = query + " datetime,";
                        break;
                    default:
                        query = query + ",";
                        break;
                }
            }
            query = query.Remove(query.Length-1,1);
            query = query + ")";
            DatabaseUtility du = new DatabaseUtility();

            du.nonQueryDb(defaultDbLoc,query);

            query = null;

            /**Inserts data from previous query into temporary TableA**/
```

```
                foreach(DataRow rows in tempDt.Rows)
                {
                        query = "insert into TableA values (";
                        foreach(DataColumn col in tempDt.Columns)
                        {
                                string data = rows[col].ToString();
                                query = query + """+data.Replace("""", """")+"',";
                        }
                        query = query.Remove(query.Length-1,1);
                        query = query + ")";
                        du.nonQueryDb(defaultDbLoc,query);
                }

        }
        else
        {
                //create tableB
                string query = "create table TableB (";

                /**Creates appropriate query depending on type**/
                foreach(DataColumn colName in tempDt.Columns)
                {
                        query = query + colName.ColumnName;

                        /**Determines column types**/
                        string type = colName.DataType.ToString();
                        switch(type)
                        {
                                case "System.String":
```

```
                    query =
                    query + "
                    nchar(50),";
                    break;
                case
                "System.Int32":
                    query =
                    query + "
                    int,";
                    break;
                case
                "System.DateTime":
                    query =
                    query + "
                    datetime,";
                    break;
                default:
                    query =
                    query + ",";
                    break;
            }
        }
        query =
        query.Remove(query.Length-
        1,1);
        query = query + ")";

        DatabaseUtility du = new
        DatabaseUtility();
        du.nonQueryDb
        (defaultDbLoc,query);
        query = null;

        /**Inserts data from previous
        query into temporary TableB**/
        foreach(DataRow rows in
        tempDt.Rows)
        {
            query = "insert into TableB
            values (";
            foreach(DataColumn col in
            tempDt.Columns)
            {
                string data =
                rows[col].ToString();
```

```
                                        query = query +
                                        """+data.Replace("""",
                                        """")+"',";
                                }
                                query                    =
                                query.Remove(query.Leng
                                th-1,1);
                                query = query + ")";
                                du.nonQueryDb
                                (defaultDbLoc,query);
                        }

                }
        }
        else
        {
                Operator opTable = getOperator(input[i]);
                tempO.setDbLoc(opTable.getDbLoc());
        }
}

/**
 * Determines appropriate query depending on type of
 * operator and type of inputs. Sends the query and
 * populates a data table.
**/
switch(tempO.getType())
{
        case "Select":
                string selectVarA = input[0];
                Operator          selectOperA          =
                getOperator(selectVarA);
                bool selTableACreated = false;
                if(selectVarA.StartsWith("Table"))
                {
                        tempO.setQuery("select distinct * from "
                        + selectOperA.getQuery());

                }
                else
                {
                        tempO.setQuery("select  distinct  *  from
                        tableA");
                        tempO.setDbLoc(defaultDbLoc);
```

```
                selTableACreated = true;

            }
            string selectVarB = input[1];
            if(selectVarB.StartsWith("Criteria"))
            {
                Operator selectOperB =
                getOperator(selectVarB);
                if(selectVarA.StartsWith("Table"))
                {
                    tempO.setQuery(tempO.getQuer
                    y()+ " where
                    "+selectOperB.getQuery());
                }
                else
                {
                    string opText =
                    selectOperB.getQuery();
                    opText = criteriaProcess(opText);
                    tempO.setQuery(tempO.getQuer
                    y()+ " where " +opText);
                }
            }
            DatabaseUtility duSelect = new
            DatabaseUtility();

            tempDt = duSelect.queryDb(tempO.getDbLoc()
            ,tempO.getQuery());

            if(selTableACreated)
            {
                string dropQuery = "drop table TableA";
                duSelect.nonQueryDb
                (defaultDbLoc,dropQuery);
            }
            break;

        case "Project":
            string projectVarB = input[1];
            Operator projectOperB =
            getOperator(projectVarB);
            bool prjTableACreated = false;
            if(projectVarB.StartsWith("Criteria"))
            {
                string projectOpText =
                projectOperB.getQuery();
```

```
                projectOpText =
                criteriaProcess(projectOpText);
                tempO.setQuery("select distinct
                "+projectOpText+ " from");

        }
        string projectVarA = input[0];
        Operator projectOperA =
        getOperator(projectVarA);
        if(projectVarA.StartsWith("Table"))
        {
                tempO.setQuery(tempO.getQuery()+"
                "+projectOperA.getQuery());
        }
        else
        {
                tempO.setQuery(tempO.getQuery()+"
                TableA");
                tempO.setDbLoc(defaultDbLoc);
                if(projectVarB != null)
                {
                        prjTableACreated = true;

                }
        }
        DatabaseUtility duProject = new
        DatabaseUtility();
        tempDt =
        duProject.queryDb(tempO.getDbLoc(),
        tempO.getQuery());
        if(prjTableACreated)
        {
                duProject.nonQueryDb(defaultDbLoc,
                "drop table TableA");
        }
        break;

case "Join":
        string joinVarA = input[0];
        Operator joinOperA = getOperator(joinVarA);
        bool joinTableACreated = false;
        bool joinTableBCreated = false;
        string joinVarB = input[1];
        Operator joinOperB = getOperator(joinVarB);
        if(joinVarA.StartsWith("Table"))
        {
```

```
            if(joinVarB.StartsWith("Table"))
            {
                tempO.setQuery("select distinct *
                from " + joinOperA.getQuery()
                +","+joinOperB.getQuery());
            }
            else
            {
                tempO.setQuery("select distinct *
                from "+joinOperA.getQuery()+",
                tableB");
                joinTableBCreated = true;
                tempO.setDbLoc(defaultDbLoc);
            }
        }
        else
        {
            if(joinVarB.StartsWith("Table"))
            {
                tempO.setQuery("select distinct *
                from tableA,"+
                joinOperB.getQuery());
            }
            else
            {
                tempO.setQuery("select distinct *
                from tableA, tableB");
                joinTableBCreated = true;
            }
            tempO.setDbLoc(defaultDbLoc);
            joinTableACreated = true;

        }
        string joinVarC = input[2];
        if(joinVarC.StartsWith("Criteria"))
        {
            Operator joinOperC =
            getOperator(joinVarC);
            string opText = joinOperC.getQuery();
            if(opText.IndexOf('=')>0 &&
            opText.Length > opText.IndexOf('=') +
            1)
            {
                string [ ] sides = opText.Split('=');
                if(joinTableACreated)
```

```
                {

                        string [] sideA =
                        sides[0].Split('.');
                        if(joinTableBCreated)
                        {
                                string [] sideB =
                                sides[1].Split('.');
                                opText =
                                "TableA."+sideA[1]+
                                "=TableB." +
                                sideB[1];

                        }
                        else
                        {
                                opText =
                                "TableA."+sideA[1]+
                                "="+ sides[1];
                        }
                }
                else
                {
                        if(joinTableBCreated)
                        {
                                string [] sideB =
                                sides[1].Split('.');
                                opText =
                                sides[0]+"=TableB."
                                + sideB[1];
                        }

                }
        }
        tempO.setQuery(tempO.getQuery() + "
        where " +opText);
}
DatabaseUtility duJoin = new DatabaseUtility();
tempDt = duJoin.queryDb(tempO.getDbLoc(),
tempO.getQuery());
if(joinTableACreated)
{
        string dropQuery = "drop table TableA";
        duJoin.nonQueryDb(defaultDbLoc,drop
        Query);
}
```

```
                if(joinTableBCreated)
                {
                        string dropQuery = "drop table TableB";
                        duJoin.nonQueryDb(defaultDbLoc,drop
                        Query);
                }
                break;

        case "Diff":
                string diffVarC = input[2];
                if(diffVarC.StartsWith("Criteria"))
                {
                        Operator diffOperC =
                        getOperator(diffVarC);
                        string opText = diffOperC.getQuery();
                        opText = criteriaProcess(opText);
                        diffVarC = opText;
                }
                string diffVarA = input[0];
                Operator diffOperA = getOperator(diffVarA);
                bool diffTableACreated = false;
                bool diffTableBCreated = false;
                string diffVarB = input[1];
                Operator diffOperB = getOperator(diffVarB);
                if(diffVarA.StartsWith("Table"))
                {
                        if(diffVarB.StartsWith("Table"))
                        {
                                tempO.setQuery("select distinct "
                                +diffVarC+ " from " +
                                diffOperA.getQuery()+" where
                                "+diffVarC+" not in(select
                                "+diffVarC+" from
                                "+diffOperB.getQuery()+")");
                        }
                        else
                        {

                                tempO.setDbLoc(defaultDbLoc);
                                if(diffVarC != null)
                                {
                                        tempO.setQuery("select
                                        distinct " +diffVarC+ " from
                                        " + diffOperA.getQuery()+"
                                        where "+diffVarC+" not
```

```
                    in(select "+diffVarC+" from
                    TableB)");
                    diffTableBCreated = true;
                }
            }
        }
        else
        {
            if(diffVarB.StartsWith("Table"))
            {
                tempO.setQuery("select distinct "
                +diffVarC+ " from TableA where
                "+diffVarC+" not in(select
                "+diffVarC+" from
                "+diffOperB.getQuery()+")");
            }
            else
            {

                if(diffVarC != null)
                {
                    tempO.setQuery("select
                    distinct " +diffVarC+ " from
                    TableA where "+diffVarC+"
                    not in(select "+diffVarC+"
                    from TableB)");
                    diffTableBCreated = true;
                }
            }
            tempO.setDbLoc(defaultDbLoc);
            if(diffVarC != null)
            {
                diffTableACreated = true;
            }

        }
        DatabaseUtility duDiff = new DatabaseUtility();
        tempDt = duDiff.queryDb(tempO.getDbLoc(),
        tempO.getQuery());
        if(diffTableACreated)
        {
            string dropQuery = "drop table TableA";
            duDiff.nonQueryDb(defaultDbLoc,
            dropQuery);
        }
        if(diffTableBCreated)
```

```
                {
                    string dropQuery = "drop table TableB";
                    duDiff.nonQueryDb(defaultDbLoc,
                    dropQuery);
                }
                break;

            case "Union":
                string unionVarA = input[0];
                Operator unionOperA =
                getOperator(unionVarA);
                bool unionTableACreated = false;
                bool unionTableBCreated = false;
                string unionVarB = input[1];
                Operator unionOperB =
                getOperator(unionVarB);
                if(unionVarA.StartsWith("Table"))
                {
                    if(unionVarB.StartsWith("Table"))
                    {
                        tempO.setQuery("select distinct *
                        from " +
                        unionOperA.getQuery()+" union
                        select distinct * from
                        "+unionOperB.getQuery());
                    }
                    else
                    {
                        tempO.setQuery("select distinct *
                        from " +
                        unionOperA.getQuery()+" union
                        select distinct * from TableB");
                        unionTableBCreated = true;
                        tempO.setDbLoc(defaultDbLoc);
                    }
                }
                else
                {
                    if(unionVarB.StartsWith("Table"))
                    {
                        tempO.setQuery("select distinct
                        from TableA union select distinct
                        * from "+unionOperB.getQuery());
                    }
                    else
                    {
```

```
                    tempO.setQuery("select distinct *
                    from TableA union select distinct
                    * from TableB");
                    unionTableBCreated = true;
                }
                tempO.setDbLoc(defaultDbLoc);
                unionTableACreated = true;

            }
            DatabaseUtility duUnion = new
            DatabaseUtility();
            tempDt = duUnion.queryDb(tempO.getDbLoc(),
            tempO.getQuery());
            if(unionTableACreated)
            {
                string dropQuery = "drop table TableA";
                duUnion.nonQueryDb(defaultDbLoc,
                dropQuery);
            }
            if(unionTableBCreated)
            {
                string dropQuery = "drop table TableB";
                duUnion.nonQueryDb(defaultDbLoc,
                dropQuery);
            }
            break;

        case "GrpCnt":
            string grpCntVarB = input[1];
            if(grpCntVarB.StartsWith("Criteria"))
            {
                Operator grpCntOperB =
                getOperator(grpCntVarB);
                string opText =
                grpCntOperB.getQuery();
                opText = criteriaProcess(opText);
                tempO.setQuery("select distinct
                "+opText+",count(*) as ");
                grpCntVarB = opText;
            }
            string grpCntVarC = input[2];
            if(grpCntVarC.StartsWith("Criteria"))
            {
                Operator grpCntOperC =
                getOperator(grpCntVarC);
```

```
                string opText = grpCntOperC.getQuery();
                opText = criteriaProcess(opText);
                tempO.setQuery(tempO.getQuery()+opText+" from ");
            }
            string grpCntVarA = input[0];
            Operator grpCntOperA = getOperator(grpCntVarA);
            bool grpCntTableACreated = false;
            if(grpCntVarA.StartsWith("Table"))
            {
                tempO.setQuery(tempO.getQuery()+grpCntOperA.getQuery()+" group by "+grpCntVarB);
            }
            else
            {
                tempO.setQuery(tempO.getQuery()+"tableA group by "+grpCntVarB);
                grpCntTableACreated = true;
                tempO.setDbLoc(defaultDbLoc);
            }
            DatabaseUtility duGrpCnt = new DatabaseUtility();
            tempDt = duGrpCnt.queryDb(tempO.getDbLoc(), tempO.getQuery());
            if(grpCntTableACreated)
            {
                string dropQuery = "drop table TableA";
                duGrpCnt.nonQueryDb(defaultDbLoc, dropQuery);
            }
            break;

        case "Table":
            DatabaseUtility duTable = new DatabaseUtility();
            if(tempO.getQuery()!=null)
            {
                string tableQry = "select * from "+tempO.getQuery();
                tempDt = duTable.queryDb(tempO.getDbLoc(), tableQry);
```

```
            }
            break;
        case "Criteria":
            DatabaseUtility duCriteria = new
            DatabaseUtility();
            if(tempO.getQuery()!=null)
            {
                DataTable tableCriteria = new
                DataTable();
                string opText = tempO.getQuery();
                DataRow criteriaRow;
                tableCriteria.Columns.Add("Criteria");
                string [] criterias = opText.Split(',');

                for(int c = 0; c<criterias.Length;c++)
                {
                    criteriaRow =
                    tableCriteria.NewRow();
                    criteriaRow["Criteria"]=criterias[c];
                    tableCriteria.Rows.Add
                    (criteriaRow);
                }
                tempDt = tableCriteria;
            }
            break;

        case "EqJoin":
            string eqJoinVarA = input[0];
            Operator eqJoinOperA =
            getOperator(eqJoinVarA);
            bool eqJoinTableACreated = false;
            bool eqJoinTableBCreated = false;
            string eqJoinVarB = input[1];
            Operator eqJoinOperB =
            getOperator(eqJoinVarB);
            if(eqJoinVarA.StartsWith("Table"))
            {
                if(eqJoinVarB.StartsWith("Table"))
                {
                    tempO.setQuery("select distinct *
                    from " + eqJoinOperA.getQuery()
                    +"," +eqJoinOperB.getQuery());
                }
                else
                {
```

```
                tempO.setQuery("select distinct *
                from "+eqJoinOperA.getQuery()
                +",tableB");
                eqJoinTableBCreated = true;
                tempO.setDbLoc(defaultDbLoc);
        }
}
else
{
        if(eqJoinVarB.StartsWith("Table"))
        {
                tempO.setQuery("select distinct *
                from tableA,"+
                eqJoinOperB.getQuery());
        }
        else
        {
                tempO.setQuery("select distinct *
                from tableA, tableB");
                eqJoinTableBCreated = true;
        }
        tempO.setDbLoc(defaultDbLoc);
        eqJoinTableACreated = true;

}
string eqJoinVarC = input[2];
if(eqJoinVarC.StartsWith("Criteria"))
{
        Operator eqJoinOperC =
        getOperator(eqJoinVarC);
        string opText =
        eqJoinOperC.getQuery();

        if(opText.IndexOf('=')>0 &&
        opText.Length > opText.IndexOf('=') +
        1)
        {
                string [] sides = opText.Split('=');
                if(eqJoinTableACreated)
                {

                        string [] sideA =
                        sides[0].Split('.');
                        if(eqJoinTableBCreated)
                        {
```

```
                        string [] sideB =
                        sides[1].Split('.');
                        opText =
                        "TableA."+sideA[1]+
                        "=TableB." +
                        sideB[1];


                }
                else
                {
                        opText =
                        "TableA."+sideA[1]+
                        "="+ sides[1];
                }
        }
        else
        {
                if(eqJoinTableBCreated)
                {

                        string [] sideB =
                        sides[1].Split('.');
                        opText =
                        sides[0]+"=TableB."
                        + sideB[1];
                }

        }
    }
    tempO.setQuery(tempO.getQuery() + "
    where " +opText);
}
DatabaseUtility duEqJoin = new
DatabaseUtility();
tempDt =
duEqJoin.queryDb(tempO.getDbLoc(),
tempO.getQuery());
if(eqJoinTableACreated)
{
    string dropQuery = "drop table TableA";
    duEqJoin.nonQueryDb(defaultDbLoc,
    dropQuery);
}
if(eqJoinTableBCreated)
{
```

```
                string dropQuery = "drop table TableB";
                duEqJoin.nonQueryDb(defaultDbLoc,
                dropQuery);
            }
            break;

        case "GrpAllSat":
            string grpAllSatVarB = input[1];
            string grpAllSatVarC = input[2];
            Operator grpAllSatOperB =
            getOperator(grpAllSatVarB);
            Operator grpAllSatOperC =
            getOperator(grpAllSatVarC);
            string grpAllSatVarA = input[0];
            Operator grpAllSatOperA =
            getOperator(grpAllSatVarA);
            bool grpAllSatTableACreated = false;
            if(grpAllSatVarA.StartsWith("Table"))
            {
                if(grpAllSatVarB.StartsWith("Criteria")&&
                grpAllSatVarC.StartsWith("Criteria"))
                {

                    tempO.setQuery("select distinct
                    "+grpAllSatOperB.getQuery()+"
                    from
                    "+grpAllSatOperA.getQuery()+ "
                    where
                    "+grpAllSatOperC.getQuery()+"
                    group by
                    "+grpAllSatOperB.getQuery());
                }
            }
            else
            {
                if(grpAllSatVarB.StartsWith("Criteria")&&
                grpAllSatVarC.StartsWith("Criteria"))
                {
                    string opTextB =
                    grpAllSatOperB.getQuery();
                    opTextB =
                    criteriaProcess(opTextB);
                    string opTextC =
                    grpAllSatOperC.getQuery();
                    opTextC =
                    criteriaProcess(opTextC);
```

```
                    tempO.setQuery("select distinct
                    "+opTextB+" from tableA where
                    "+opTextC+" group by
                    "+opTextB);
                    grpAllSatTableACreated = true;
                    tempO.setDbLoc(defaultDbLoc);
                }
            }
            DatabaseUtility duGrpAllSat = new
            DatabaseUtility();
            tempDt =
            duGrpAllSat.queryDb(tempO.getDbLoc(),
            tempO.getQuery());
            if(grpAllSatTableACreated)
            {
                string dropQuery = "drop table TableA";
                duGrpAllSat.nonQueryDb(defaultDbLoc,
                dropQuery);
            }
            break;

        case "GrpNSat":
            string grpNSatVarB = input[1];
            string grpNSatVarC = input[2];
            string grpNSatVarD = input[3];
            Operator grpNSatOperB =
            getOperator(grpNSatVarB);
            Operator grpNSatOperC =
            getOperator(grpNSatVarC);
            Operator grpNSatOperD =
            getOperator(grpNSatVarD);
            string grpNSatVarA = input[0];
            Operator grpNSatOperA =
            getOperator(grpNSatVarA);
            bool grpNSatTableACreated = false;
            if(grpNSatVarA.StartsWith("Table"))
            {
                if(grpNSatVarB.StartsWith("Criteria")&&
                grpNSatVarC.StartsWith("Criteria")&&
                grpNSatVarD.StartsWith("Criteria"))
                {
                    tempO.setQuery("select distinct
                    "+grpNSatOperB.getQuery()+"
                    from
                    "+grpNSatOperA.getQuery()+ "
                    where
```

```
                "+grpNSatOperC.getQuery()+"
                group by
                "+grpNSatOperB.getQuery()+"
                having
                count(*)"+grpNSatOperD.getQuer
                y());
        }
}
else
{
        if(grpNSatVarB.StartsWith("Criteria")&&
        grpNSatVarC.StartsWith("Criteria")&&
        grpNSatVarD.StartsWith("Criteria"))
        {
                string opTextB =
                grpNSatOperB.getQuery();
                opTextB =
                criteriaProcess(opTextB);
                string opTextC =
                grpNSatOperC.getQuery();
                opTextC =
                criteriaProcess(opTextC);
                string opTextD =
                grpNSatOperD.getQuery();
                opTextD =
                criteriaProcess(opTextD);
                tempO.setQuery("select distinct
                "+opTextB+" from tableA where
                "+opTextC+" group by
                "+opTextB+" having
                count(*)"+opTextD);
                grpAllSatTableACreated = true;
                tempO.setDbLoc(defaultDbLoc);
        }
}
DatabaseUtility duGrpNSat = new
DatabaseUtility();
tempDt=duGrpNSat.queryDb
(tempO.getDbLoc(),tempO.getQuery());
if(grpNSatTableACreated)
{
        string dropQuery = "drop table TableA";
        duGrpNSat.nonQueryDb(defaultDbLoc,
        dropQuery);
}
break;
```

```
case "Intersect":
        string intersectVarC = input[2];
        if(intersectVarC.StartsWith("Criteria"))
        {
                Operator intersectOperC =
                getOperator(intersectVarC);
                string opText =
                intersectOperC.getQuery();
                opText = criteriaProcess(opText);
                intersectVarC = opText;
        }
        string intersectVarA = input[0];
        Operator intersectOperA =
        getOperator(intersectVarA);
        bool intersectTableACreated = false;
        bool intersectTableBCreated = false;
        string intersectVarB = input[1];
        Operator intersectOperB =
        getOperator(intersectVarB);
        if(intersectVarA.StartsWith("Table"))
        {
                if(intersectVarB.StartsWith("Table"))
                {
                        tempO.setQuery("select distinct "
                        +intersectVarC+ " from " +
                        intersectOperA.getQuery()+"
                        where "+intersectVarC+" in(select
                        "+intersectVarC+" from
                        "+intersectOperB.getQuery()+")");
                }
                else
                {
                        tempO.setQuery("select distinct "
                        +intersectVarC+ " from " +
                        intersectOperA.getQuery()+"
                        where "+intersectVarC+" in(select
                        "+intersectVarC+" from TableB)");
                        intersectTableBCreated = true;
                        tempO.setDbLoc(defaultDbLoc);
                }
        }
        else
        {
                if(intersectVarB.StartsWith("Table"))
                {
```

```
                    tempO.setQuery("select distinct "
                    +intersectVarC+ " from TableA
                    where "+intersectVarC+" in(select
                    "+intersectVarC+" from
                    "+intersectOperB.getQuery()+")");
                }
                else
                {
                    tempO.setQuery("select distinct "
                    +intersectVarC+ " from TableA
                    where "+intersectVarC+" in(select
                    "+intersectVarC+" from TableB)");
                    intersectTableBCreated = true;
                }
                tempO.setDbLoc(defaultDbLoc);
                intersectTableACreated = true;

            }
            DatabaseUtility duIntersect = new
            DatabaseUtility();
            tempDt =
            duIntersect.queryDb(tempO.getDbLoc(),
            tempO.getQuery());
            if(intersectTableACreated)
            {
                string dropQuery = "drop table TableA";
                duIntersect.nonQueryDb(defaultDbLoc,
                dropQuery);
            }
            if(intersectTableBCreated)
            {
                string dropQuery = "drop table TableB";
                duIntersect.nonQueryDb
                (defaultDbLoc,dropQuery);
            }
            break;

        case "GrpMin":
            string grpMinVarB = input[1];
            string grpMinVarC = input[2];
            Operator grpMinOperB =
            getOperator(grpMinVarB);
            Operator grpMinOperC =
            getOperator(grpMinVarC);
            string grpMinVarA = input[0];
```

```
Operator grpMinOperA = getOperator(grpMinVarA);
bool grpMinTableACreated = false;
if(grpMinVarA.StartsWith("Table"))
{
	if(grpMinVarB.StartsWith("Criteria")&& grpMinVarC.StartsWith("Criteria"))
	{
		string opTextC = grpMinOperC.getQuery();
		opTextC = criteriaProcess(opTextC);
		tempO.setQuery("select distinct "+grpMinOperB.getQuery()+", min("+grpMinOperC.getQuery()+") as MIN_"+opTextC+" from "+grpMinOperA.getQuery()+" group by "+grpMinOperB.getQuery());
	}
}
else
{
	if(grpMinVarB.StartsWith("Criteria")&& grpMinVarC.StartsWith("Criteria"))
	{
		string opTextB = grpMinOperB.getQuery();
		opTextB = criteriaProcess(opTextB);
		string opTextC = grpMinOperC.getQuery();
		opTextC = criteriaProcess(opTextC);
		tempO.setQuery("select distinct "+opTextB+" ,min("+opTextC+") as MIN_"+opTextC+" from tableA group by "+opTextB);
		grpMinTableACreated = true;
		tempO.setDbLoc(defaultDbLoc);
	}
}
DatabaseUtility duGrpMin = new DatabaseUtility();
```

```
                tempDt =
                duGrpMin.queryDb(tempO.getDbLoc(),
                tempO.getQuery());
                if(grpMinTableACreated)
                {
                        string dropQuery = "drop table TableA";
                        duGrpMin.nonQueryDb(defaultDbLoc,
                        dropQuery);
                }
                break;

        case "GrpMax":
                string grpMaxVarB = input[1];
                string grpMaxVarC = input[2];
                Operator grpMaxOperB =
                getOperator(grpMaxVarB);
                Operator grpMaxOperC =
                getOperator(grpMaxVarC);
                string grpMaxVarA = input[0];
                Operator grpMaxOperA =
                getOperator(grpMaxVarA);
                bool grpMaxTableACreated = false;
                if(grpMaxVarA.StartsWith("Table"))
                {
                        if(grpMaxVarB.StartsWith("Criteria")&&
                        grpMaxVarC.StartsWith("Criteria"))
                        {
                                string opTextC =
                                grpMaxOperC.getQuery();
                                opTextC =
                                criteriaProcess(opTextC);
                                tempO.setQuery("select distinct
                                "+grpMaxOperB.getQuery()+",
                                max("+grpMaxOperC.getQuery()
                                +") as MAX_"+opTextC+" from
                                "+grpMaxOperA.getQuery()+"
                                group by
                                "+grpMaxOperB.getQuery());
                        }
                }
                else
                {
                        if(grpMaxVarB.StartsWith("Criteria")&&
                        grpMaxVarC.StartsWith("Criteria"))
                        {
```

```
                    string opTextB =
                    grpMaxOperB.getQuery();
                    opTextB                    =
                    criteriaProcess(opTextB);
                    string          opTextC    =
                    grpMaxOperC.getQuery();
                    opTextC                    =
                    criteriaProcess(opTextC);
                    tempO.setQuery("select distinct
                    "+opTextB+" ,max("+opTextC+")
                    as MAX_"+opTextC+" from
                    tableA group by "+opTextB);
                    grpMaxTableACreated = true;
                    tempO.setDbLoc(defaultDbLoc);
                }
            }
            DatabaseUtility duGrpMax = new
            DatabaseUtility();
            tempDt                             =
            duGrpMax.queryDb(tempO.getDbLoc(),
            tempO.getQuery());
            if(grpMaxTableACreated)
            {
                string dropQuery = "drop table TableA";
                duGrpMax.nonQueryDb(defaultDbLoc,
                dropQuery);
            }
            break;

        case "GrpAvg":
            string grpAvgVarB = input[1];
            string grpAvgVarC = input[2];
            Operator          grpAvgOperB      =
            getOperator(grpAvgVarB);
            Operator          grpAvgOperC      =
            getOperator(grpAvgVarC);
            string grpAvgVarA = input[0];
            Operator          grpAvgOperA      =
            getOperator(grpAvgVarA);
            bool grpAvgTableACreated = false;
            if(grpAvgVarA.StartsWith("Table"))
            {
                if(grpAvgVarB.StartsWith("Criteria")&&
                grpAvgVarC.StartsWith("Criteria"))
                {
```

```
                string opTextC =

                grpAvgOperC.getQuery();
                opTextC =
                criteriaProcess(opTextC);
                tempO.setQuery("select distinct
                "+grpAvgOperB.getQuery()+",
                avg("+grpAvgOperC.getQuery()+"
                ) as AVG_"+opTextC+" from
                "+grpAvgOperA.getQuery()+"
                group by
                "+grpAvgOperB.getQuery());
        }
}
else
{
        if(grpAvgVarB.StartsWith("Criteria")&&
        grpAvgVarC.StartsWith("Criteria"))
        {
                string opTextB =
                grpAvgOperB.getQuery();
                opTextB =
                criteriaProcess(opTextB);
                string opTextC =
                grpAvgOperC.getQuery();
                opTextC =
                criteriaProcess(opTextC);
                tempO.setQuery("select distinct
                "+opTextB+" ,avg("+opTextC+")
                as AVG_"+opTextC+" from
                tableA group by "+opTextB);
                grpAvgTableACreated = true;
                tempO.setDbLoc(defaultDbLoc);
        }
}
DatabaseUtility duGrpAvg = new
DatabaseUtility();
tempDt =
duGrpAvg.queryDb(tempO.getDbLoc(),
tempO.getQuery());
if(grpAvgTableACreated)
{
        string dropQuery = "drop table TableA";
        duGrpAvg.nonQueryDb(defaultDbLoc,
        dropQuery);
}
```

```
                    break;

                default:
                    break;

        }
        return tempDt;
}

/**
 * Processes criteria to remove original table
 * information when temporary tables are to be
 * used.
**/
private string criteriaProcess(string criteria)
{
        string text = criteria;
        if(criteria != null)
        {
            int position = 0;
            while(text.IndexOfAny(new   char[]{'.',',','   ','=','!','<','>'},
            position)>0)
            {
                int     newPosition     =     text.IndexOfAny(new
                char[]{'.',',',' ','=','!','<','>'},position);
                if(text.IndexOf('.')==newPosition)
                {
                    text                                        =
                    text.Remove(position,newPosition-
                    position+1);
                    position = 0;
                }
                else
                {
                    position = newPosition + 1;
                }
            }
        }
        return text;
}

/**
 * Determines which part of the operator a point is at.
 * Method returns the location as a string.
 * (i.e. space, body, move, delete, inputA, inputB
 * inputC, inputD, inputE, or output)
```

```
**/
public string clickedAt(Point p)
{
        int operatorLocation = location(p);
        string result = "space";

        /**If the given point is occupied by an operator**/
        if (operatorLocation >= 0)
        {
                result = "body";
                Point opPoint = new Point(0,0);
                op = (Operator) operatorList[operatorLocation];
                opPoint = op.getPoint();

                /**
                 * If point lies within this space then move node
                 * was clicked.  Remove the operator from the list
                 * and work with temporary operator instead.
                **/
                if (opPoint.X<=p.X && opPoint.X+5>=p.X)
                {
                        if (opPoint.Y<=p.Y && opPoint.Y+5>=p.Y)
                        {
                                result = "move";
                                remove(operatorLocation);
                        }
                }

                /**
                 * If point lies within this space then delete
                 * node was clicked.  Remove the operator from
                 * the list and update other operators connected
                 * to it.
                **/
                if (opPoint.X+50<=p.X && opPoint.X+55>=p.X)
                {
                        if (opPoint.Y<=p.Y && opPoint.Y+5>=p.Y)
                        {
                                result = "delete";
                                remove(operatorLocation);
                                string find = op.getName();
                                Operator opTemp = new Operator();
                                for (int e=0;e!=operatorList.Count;e++)
                                {
                                        opTemp          =          (Operator)
                                        operatorList[e];
```

```
string [] inputTemp = opTemp.getInput();
for (int g=0; g! = opTemp.getNumberOfInputNodes(); g++)
{
    if (inputTemp[g] == find)
    {
        if(opTemp.getNumberOfInputNodes() == 2)
        {
            if(g == 0)
            {
                inputTemp[g] = "inputA";
            }
            else
            {
                inputTemp[g] = "inputE";
            }
        }
        if(opTemp.getNumberOfInputNodes() == 3)
        {
            if(g == 0)
            {
                inputTemp[g] = "inputA";
            }
            else
            {
                if(g == 1)
                {
                    inputTemp[g] = "inputC";
                }
                if(g == 2)
                {
```

```
                                    inputTemp [g] = "inputE";
                                }
                            }
                        }
                        if(opTemp.getNumberOfInputNodes() == 4)
                        {
                            if(g == 0)
                            {
                                inputTemp[g] = "inputA";
                            }
                            else
                            {
                                if(g == 1)
                                {
                                    inputTemp[g] = "inputB";
                                }
                                if(g == 2)
                                {
                                    inputTemp[g] = "inputD";
                                }
                                if(g == 3)
                                {
                                    inputTemp[g] = "inputE";
                                }
                            }
                        }
                        opTemp.setInput(inputTemp);
                        break;
                    }
                }
            }

        }
    }

    /**
     * If point lies within this space then output
```

```
 * node was clicked.
**/
if (opPoint.X+25<=p.X && opPoint.X+30>=p.X)
{
        if (opPoint.Y+25<=p.Y && opPoint.Y+30>=p.Y)
        {
                result = "output";
        }
}

/**
 * If point lies within this space then inputA
 * node was clicked.
**/
if (opPoint.X+10<=p.X && opPoint.X+15>=p.X)
{
        if (opPoint.Y-5<=p.Y && opPoint.Y>=p.Y)
        {
                result = "inputA";
        }
}

/**
 * If point lies within this space and operator
 * type is GrpNSat then inputB node was
 * clicked.
**/
if (opPoint.X+20<=p.X && opPoint.X+25>=p.X)
{
        if (opPoint.Y-5<=p.Y && opPoint.Y>=p.Y)
        {
                string type = op.getType();
                if (type == "GrpNSat")
                {
                        result = "inputB";
                }
        }
}

/**
 * If point lies within this space and operator
 * type is not GrpNSat then inputC node
 * was clicked.
**/
if (opPoint.X+25<=p.X && opPoint.X+30>=p.X)
{
```

```
                    if (opPoint.Y-5<=p.Y && opPoint.Y>=p.Y)
                    {
                        string type = op.getType();
                        if (type == "GrpNSat")
                        {
                        }
                        else
                        {
                            result = "inputC";
                        }
                    }
                }

                /**
                 * If point lies within this space and type
                 * of operator is GrpNSat then inputD was
                 * clicked.
                **/
                if (opPoint.X+30<=p.X && opPoint.X+35>=p.X)
                {
                    if (opPoint.Y-5<=p.Y && opPoint.Y>=p.Y)
                    {
                        string type = op.getType();
                        if (type == "GrpNSat")
                        {
                            result = "inputD";
                        }
                    }
                }

                /**
                 * If point lies within this space then inputE
                 * was clicked.
                **/
                if (opPoint.X+40<=p.X && opPoint.X+45>=p.X)
                {
                    if (opPoint.Y-5<=p.Y && opPoint.Y>=p.Y)
                    {
                        result = "inputE";
                    }
                }
                if (result == "body")
                {
                }
            }
            return result;
```

```
            }//end of clickedAt
        }//end of OperatorUtility
}
```

7. DatabaseUtility.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Collections;
using System.Data;
using System.Data.SqlServerCe;

namespace DFQL
{
        /**
         * Utility for accessing databases.  Gets database information and
         * used for passing SQL queries and non-queries.
        **/
        public class DatabaseUtility
        {
                /**
                 * DatabaseUtility constructor
                **/
                public DatabaseUtility()
                {
                        //
                        // TODO: Add constructor logic here
                        //
                }

                /**Gets database name given filepath**/
                public string getDatabaseName(String filePath)
                {
                        string databaseName = Path.GetFileNameWithoutExtension
                        (filePath);
                        return databaseName;
                }

                /**Gets table names in arraylist given filepath**/
                public ArrayList getTableNames(String filePath)
                {
                        string strGetTableNames =
                                "SELECT TABLE_NAME" +
                                "  FROM Information_Schema.TABLES" +
                                " WHERE TABLE_TYPE = 'TABLE'";
                        ArrayList tableArray = new ArrayList();
                        string  strConn = "Data Source=" + filePath;
                        SqlCeConnection connDB = new SqlCeConnection
                        (strConn);
                        try
```

```
        {
            connDB.Open();
        }
        catch ( SqlCeException exSQL )
        {
            MessageBox.Show("Unable to open database at " +
                strConn +
                ". Reason: " +
                exSQL.Errors[0].Message);
        }
        SqlCeDataReader drdrTableNames;
        SqlCeCommand cmndDB = new SqlCeCommand
        (strGetTableNames, connDB);
        drdrTableNames = cmndDB.ExecuteReader();
        while ( drdrTableNames.Read() )
        {
            tableArray.Add(drdrTableNames.GetString(0));
        }
        drdrTableNames.Close();
        connDB.Close();
        return tableArray;
}

/**Gets column names is arraylist given filepath and table name**/
public ArrayList getColumnNames(string filePath, string tableName)
{
        string strGetColumnNames = "SELECT * FROM " +
        tableName;
        ArrayList columnArray = new ArrayList();
        string  strConn = "Data Source=" + filePath;
        SqlCeConnection connDB = new
        SqlCeConnection(strConn);
        try
        {
            connDB.Open();
        }
        catch ( SqlCeException exSQL )
        {
            MessageBox.Show("Unable to open database at " +
                strConn +
                ". Reason: " +
                exSQL.Errors[0].Message);
        }
        SqlCeDataReader drdrColumnNames;
```

```
        SqlCeCommand cmndDB = new
        SqlCeCommand(strGetColumnNames, connDB);

        drdrColumnNames = cmndDB.ExecuteReader();
        for ( int c = 0; c < drdrColumnNames.FieldCount; c++ )
        {
                columnArray.Add(drdrColumnNames.GetName(c));
        }
        drdrColumnNames.Close();
        connDB.Close();
        return columnArray;
}

/**
 * Queries database given filepath and query and returns
 * result as datatable
**/
public DataTable queryDb (string filePath, string query)
{
        DataTable tblResult = new DataTable("Result");
        if(filePath != null && query != null)
        {
                string  strConn = "Data Source=" + filePath;

                SqlCeConnection connDB = new SqlCeConnection
                (strConn);
                try
                {

                        connDB.Open();

                }
                catch ( SqlCeException exSQL )
                {
                        MessageBox.Show ("Query error. Unable to
                        open database at " +
                                strConn +
                                ". Reason: " +
                                exSQL.Errors[0].Message);
                }
                SqlCeDataReader reader;
                SqlCeCommand  cmndDB = new SqlCeCommand
                (query, connDB);
                try
                {
                        reader = cmndDB.ExecuteReader();
```

```
                int colCount = 0;
                colCount = reader.FieldCount;

                string [] colName = new string[colCount];
                int dupCount = 0;
                for ( int c = 0; c < colCount; c++ )
                {
                        colName[c] = reader.GetName(c);
                        for(int d = 0; d < c; d++)
                        {
                                if(colName[c]==colName[d])
                                {
                                        dupCount++;
                                        colName[c] = colName[c]
                                        + dupCount.ToString();
                                }
                        }
                        tblResult.Columns.Add(colName[c],
                        reader.GetFieldType(c));
                }

                while (reader.Read())
                {
                        DataRow rowData;
                        rowData = tblResult.NewRow();
                        for ( int r = 0; r < colCount; r++ )
                        {
                                rowData[colName[r]]             =
                                reader.GetValue(r);
                        }
                        tblResult.Rows.Add(rowData);
                }
                reader.Close();
        }
        catch (SqlCeException exSQL )
        {
                MessageBox.Show("Something wrong with
                SQL statement: " +
                        query +
                        ". Reason: " +
                        exSQL.Errors[0].Message);
        }
        connDB.Close();
}
return tblResult;
```

```
        }

        /**Sends non-query SQL to database given filepath and query**/
        public void nonQueryDb (string filePath, string query)
        {
            string  strConn = "Data Source=" + filePath;
            if (filePath!=null && query !=null)
            {
                SqlCeConnection connDB = new
                SqlCeConnection(strConn);
                try
                {
                    connDB.Open();
                    SqlCeCommand cmndDB = new
                    SqlCeCommand(query, connDB);
                    cmndDB.ExecuteNonQuery();;

                }
                catch ( SqlCeException exSQL )
                {
                    MessageBox.Show("Non-Query error. Unable
                    to open database at " +
                        strConn +
                        ". Reason: " +
                        exSQL.Errors[0].Message);
                }
                connDB.Close();
            }
        }
    }
}
```

APPENDIX – B

LOCAL DATABASE (ENROLLMENT)

Course:

CNO	TITLE	INO
C001	Course1	I001
C002	Course2	I002
C003	Course3	I001
C004	Course4	I002

Enroll:

ECno	ESno	GRADE
C001	111-11-1111	A
C001	111-11-1112	B
C001	111-11-1113	B
C001	111-11-1114	A
C001	111-11-1115	B
C001	111-11-1116	A
C002	111-11-1111	A
C002	111-11-1112	A
C003	111-11-1113	A
C003	111-11-1114	A
C003	111-11-1115	B

C003	111-11-1116	B
C004	111-11-1112	A

Instructor:

INO	INAME
I001	Instructor01
I002	Instructor02
I003	Instructor03

Student:

SNO	SNAME	AGE	GENDER
111-11-1111	Student01	22	female
111-11-1112	Student02	25	male
111-11-1113	Student03	31	female
111-11-1114	Student04	38	male
111-11-1115	Student05	27	female
111-11-1116	Student06	29	male

REMOTE DATABASE (NORTHWIND)

Categories:

Data in Table 'Categories' in 'Northwind'

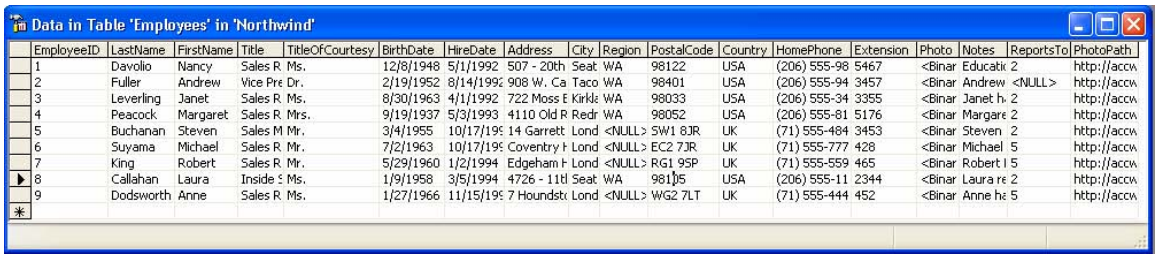
CategoryID	CategoryName	Description	Picture
1	Beverages	Soft drinks, coffees, teas, beers, and ales	<Binary>
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings	<Binary>
3	Confections	Desserts, candies, and sweet breads	<Binary>
4	Dairy Products	Cheeses	<Binary>
5	Grains/Cereals	Breads, crackers, pasta, and cereal	<Binary>
6	Meat/Poultry	Prepared meats	<Binary>
7	Produce	Dried fruit and bean curd	<Binary>
8	Seafood	Seaweed and fish	<Binary>

Customers:

Data in Table 'Customers' in 'Northwind'

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Represent	Obere Str. 57	Berlin	<NULL>	12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillc Empare	Ana Trujillo	Owner	Avda. de la C	México D.F	<NULL>	05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Ta	Antonio Moreno	Owner	Mataderos 2	México D.F	<NULL>	05023	Mexico	(5) 555-3932	<NULL>
AROUT	Around the Horn	Thomas Hardy	Sales Represent	120 Hanover	London	<NULL>	WA1 1DP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabbköp	Christina Berglur	Order Administr	Berguvsväger	Luleå	<NULL>	S-958 22	Sweden	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikate	Hanna Moos	Sales Represent	Forsterstr. 57	Mannheim	<NULL>	68306	Germany	0621-08460	0621-08924
BLONP	Blondesddsl père el	Frédérique Citea	Marketing Mana	24, place Kléb	Strasbourg	<NULL>	67000	France	88.60.15.31	88.60.15.32
BOLID	Bólido Comidas pre	Martín Sommer	Owner	C/ Araquil, 67	Madrid	<NULL>	28023	Spain	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app'	Laurence Lebiha	Owner	12, rue des Bc	Marseille	<NULL>	13008	France	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dolar Mark	Elizabeth Lincoln	Accounting Man	23 Tsawasser	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745
BSBEV	B's Beverages	Victoria Ashwort	Sales Represent	Fauntleroy Cir	London	<NULL>	EC2 5NT	UK	(171) 555-1212	<NULL>
CACTU	Cactus Comidas pa	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aire	<NULL>	1010	Argentina	(1) 135-5555	(1) 135-4892
CENTC	Centro comercial M	Francisco Chanç	Marketing Mana	Sierras de Gra	México D.F	<NULL>	05022	Mexico	(5) 555-3392	(5) 555-7293
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	<NULL>	3012	Switzerlan	0452-076545	<NULL>
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíac	Sao Paulo	SP	05432-043	Brazil	(11) 555-7647	<NULL>
CONSH	Consolidated Holdir	Elizabeth Brown	Sales Represent	Berkeley Gard	London	<NULL>	WX1 6LT	UK	(171) 555-2282	(171) 555-9199
DRACD	Drachenblut Delikat	Sven Ottlieb	Order Administr	Walserweg 21	Aachen	<NULL>	52066	Germany	0241-039123	0241-059428
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Ci	Nantes	<NULL>	44000	France	40.67.88.88	40.67.89.89
EASTC	Eastern Connectior	Ann Devon	Sales Agent	35 King Georg	London	<NULL>	WX3 6FW	UK	(171) 555-0297	(171) 555-3373
ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6	Graz	<NULL>	8010	Austria	7675-3425	7675-3426
FAMIA	Familia Arquibaldo	Aria Cruz	Marketing Assist	Rua Orós, 92	Sao Paulo	SP	05442-030	Brazil	(11) 555-9857	<NULL>
FISSA	FISSA Fabrica Inter	Diego Roel	Accounting Man	C/ Moralzarza	Madrid	<NULL>	28034	Spain	(91) 555 94 44	(91) 555 55 93
FOLIG	Folies gourmandes	Martine Rancé	Assistant Sales	184, chaussé	Lille	<NULL>	59000	France	20.16.10.16	20.16.10.17
FOLKO	Folk och fä HB	Maria Larsson	Owner	Åkergatan 24	Bräcke	<NULL>	S-844 67	Sweden	0695-34 67 21	<NULL>
FRANK	Frankenversand	Peter Franken	Marketing Mana	Berliner Platz	München	<NULL>	80805	Germany	089-0877310	089-0877451
FRANR	France restauratior	Carine Schmitt	Marketing Mana	54, rue Royal	Nantes	<NULL>	44000	France	40.32.21.21	40.32.21.20
FRANS	Franchi S.p.A.	Paolo Accorti	Sales Represent	Via Monte Bian	Torino	<NULL>	10100	Italy	011-4988260	011-4988261
FURIB	Furia Bacalhau e Fr	Lino Rodriguez	Sales Manager	Jardim das ro	Lsboa	<NULL>	1675	Portugal	(1) 354-2534	(1) 354-2535
GALED	Galería del gastróno	Eduardo Saavec	Marketing Mana	Rambla de Ca	Barcelona	<NULL>	08022	Spain	(93) 203 4560	(93) 203 4561
GODOS	Godos Cocna Típica	José Pedro Frey	Sales Manager	C/ Romero, 3	Sevilla	<NULL>	41101	Spain	(95) 555 82 82	<NULL>
GOURL	Gourmet Lanchonet	André Fonseca	Sales Associate	Av. Brasil, 44	Campinas	SP	04876-786	Brazil	(11) 555-9482	<NULL>
GREAL	Great Lakes Food M	Howard Snyder	Marketing Mana	2732 Baker Bl	Eugene	OR	97403	USA	(503) 555-7555	<NULL>
GROSR	GROSELLA-Restaur	Manuel Pereira	Owner	5ª Ave. Los P.	Caracas	DF	1081	Venezuela	(2) 283-2951	(2) 283-3397
HANAR	Hanari Carnes	Mario Pontes	Accounting Man	Rua do Paço,	Rio de Jane	RJ	05454-876	Brazil	(21) 555-0091	(21) 555-8765
HILAA	HILARION-Abastos	Carlos Hernánde	Sales Represent	Carrera 22 co	San Cristób	Táchira	5022	Venezuela	(5) 555-1340	(5) 555-1948
HUNGC	Hungry Coyote Imp	Yoshi Latimer	Sales Represent	City Center Pl	Elgin	OR	97827	USA	(503) 555-6874	(503) 555-2376
HUNGO	Hungry Owl All-Nigh	Patricia McKenna	Sales Associate	8 Johnstown F	Cork	Co. Cork	<NULL>	Ireland	2967 542	2967 3333
ISLAT	Island Trading	Helen Bennett	Marketing Mana	Garden House	Cowes	Isle of Wight	PO31 7PJ	UK	(198) 555-8888	<NULL>
KOENE	Königlich Essen	Philip Cramer	Sales Associate	Maubelstr. 90	Brandenbur	<NULL>	14776	Germany	0555-09876	<NULL>

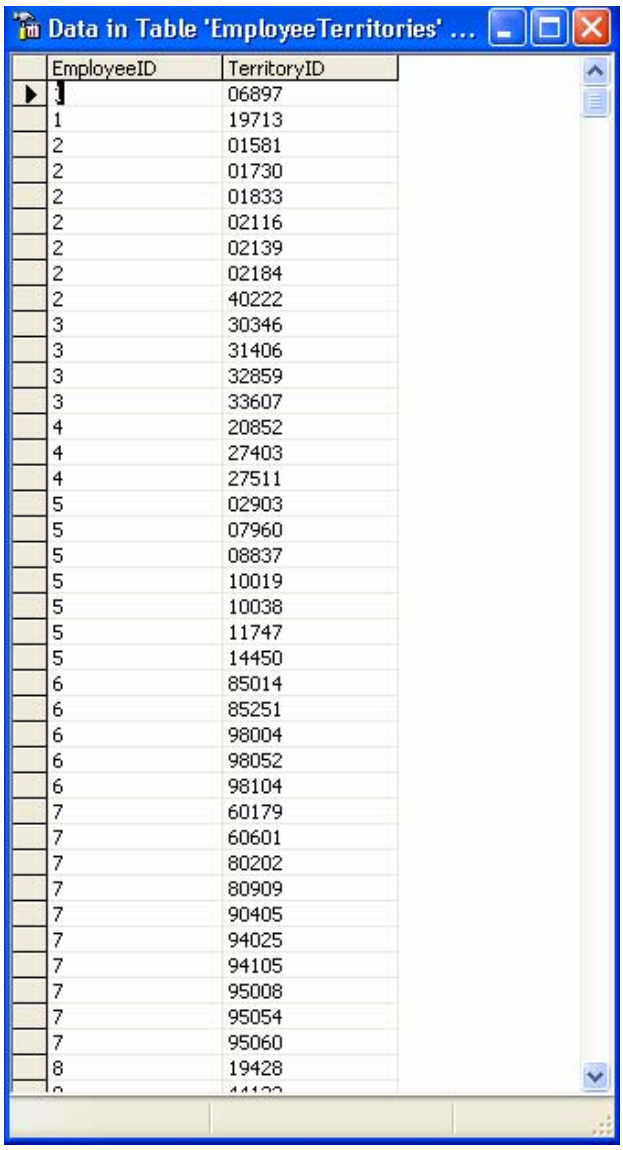
Employees:



Data in Table 'Employees' in 'Northwind'

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region	PostalCode	Country	HomePhone	Extension	Photo	Notes	ReportsTo	PhotoPath
1	Davolio	Nancy	Sales R	Ms.	12/8/1948	5/1/1992	507 - 20th	Seat	WA	98122	USA	(206) 555-98	5467	<Binar	Educati	2	http://acc
2	Fuller	Andrew	Vice Pre	Dr.	2/19/1952	8/14/1992	908 W. Ca	Taco	WA	98401	USA	(206) 555-94	3457	<Binar	Andrew	<NULL>	http://acc
3	Leverling	Janet	Sales R	Ms.	8/30/1963	4/1/1992	722 Moss E	Kirkl	WA	98033	USA	(206) 555-34	3355	<Binar	Janet h	2	http://acc
4	Peacock	Margaret	Sales R	Mrs.	9/19/1937	5/3/1993	4110 Old R	Redr	WA	98052	USA	(206) 555-81	5176	<Binar	Margare	2	http://acc
5	Buchanan	Steven	Sales M	Mr.	3/4/1955	10/17/199	14 Garrett	Lond	<NULL>	SW1 8JR	UK	(71) 555-484	3453	<Binar	Steven	2	http://acc
6	Suyama	Michael	Sales R	Mr.	7/2/1963	10/17/199	Coventry H	Lond	<NULL>	EC2 7JR	UK	(71) 555-777	428	<Binar	Michael	5	http://acc
7	King	Robert	Sales R	Mr.	5/29/1960	1/2/1994	Edgeham H	Lond	<NULL>	RG1 9SP	UK	(71) 555-559	465	<Binar	Robert I	5	http://acc
8	Callahan	Laura	Inside S	Ms.	1/9/1958	3/5/1994	4726 - 11t	Seat	WA	98105	USA	(206) 555-11	2344	<Binar	Laura re	2	http://acc
9	Dodsworth	Anne	Sales R	Ms.	1/27/1966	11/15/199	7 Houndst	Lond	<NULL>	WG2 7LT	UK	(71) 555-444	452	<Binar	Anne ha	5	http://acc

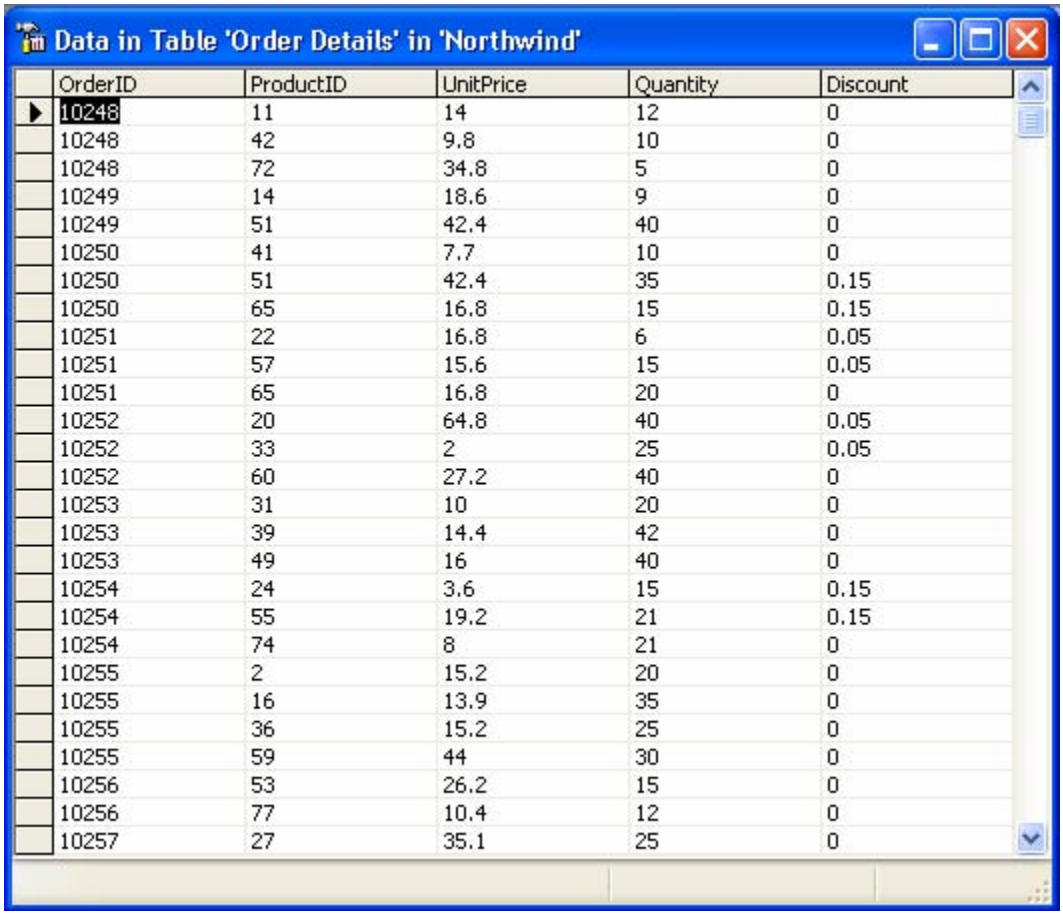
EmployeeTerritories:



Data in Table 'EmployeeTerritories' ...

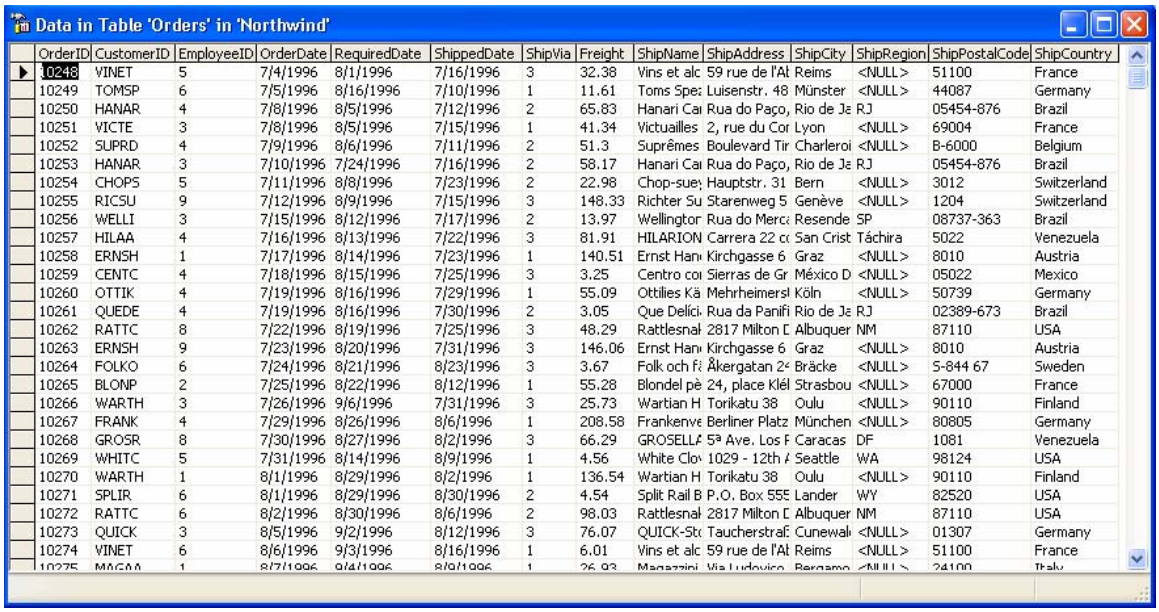
EmployeeID	TerritoryID
1	06897
1	19713
2	01581
2	01730
2	01833
2	02116
2	02139
2	02184
2	40222
3	30346
3	31406
3	32859
3	33607
4	20852
4	27403
4	27511
5	02903
5	07960
5	08837
5	10019
5	10038
5	11747
5	14450
6	85014
6	85251
6	98004
6	98052
6	98104
7	60179
7	60601
7	80202
7	80909
7	90405
7	94025
7	94105
7	95008
7	95054
7	95060
8	19428

Order Details:



OrderID	ProductID	UnitPrice	Quantity	Discount
10248	11	14	12	0
10248	42	9.8	10	0
10248	72	34.8	5	0
10249	14	18.6	9	0
10249	51	42.4	40	0
10250	41	7.7	10	0
10250	51	42.4	35	0.15
10250	65	16.8	15	0.15
10251	22	16.8	6	0.05
10251	57	15.6	15	0.05
10251	65	16.8	20	0
10252	20	64.8	40	0.05
10252	33	2	25	0.05
10252	60	27.2	40	0
10253	31	10	20	0
10253	39	14.4	42	0
10253	49	16	40	0
10254	24	3.6	15	0.15
10254	55	19.2	21	0.15
10254	74	8	21	0
10255	2	15.2	20	0
10255	16	13.9	35	0
10255	36	15.2	25	0
10255	59	44	30	0
10256	53	26.2	15	0
10256	77	10.4	12	0
10257	27	35.1	25	0

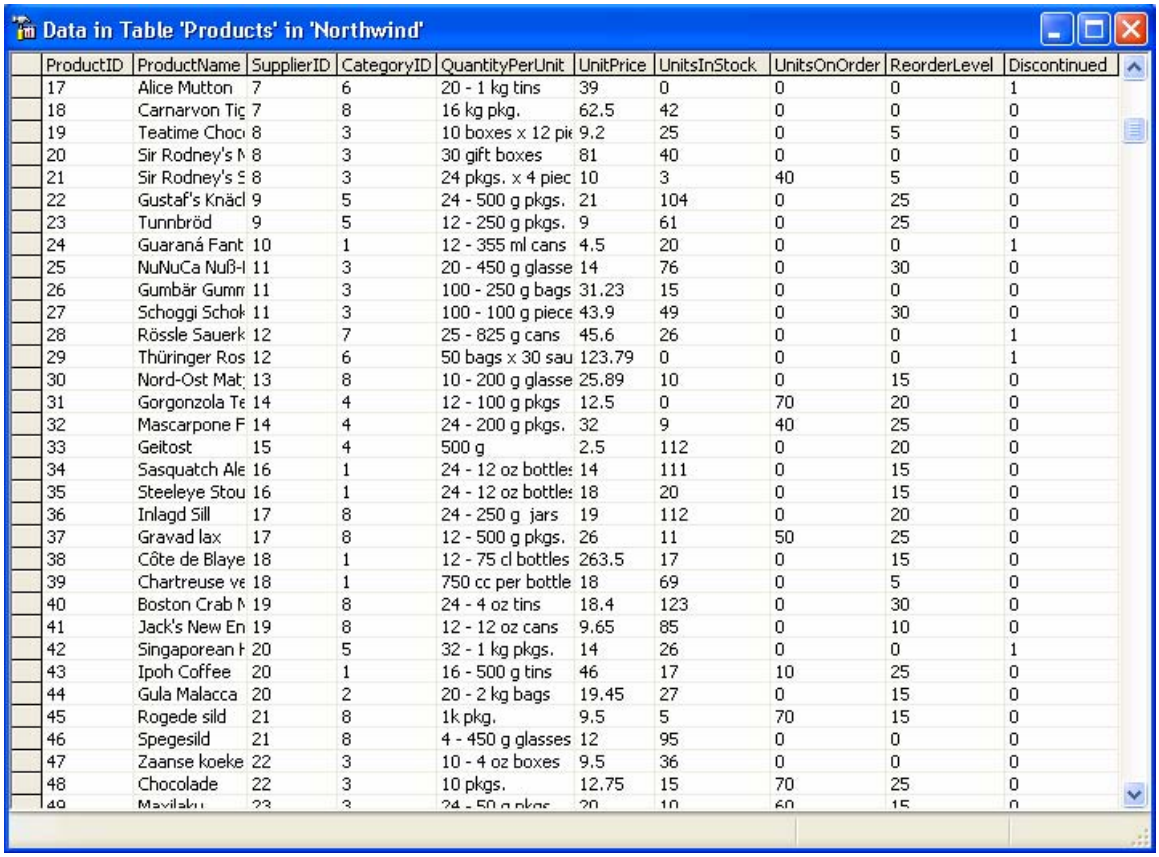
Orders:



Data in Table 'Orders' in 'Northwind'

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry
10248	VINET	5	7/4/1996	8/1/1996	7/16/1996	3	32.38	Vins et alc	59 rue de l'Al	Reims	<NULL>	51100	France
10249	TOMSP	6	7/5/1996	8/16/1996	7/10/1996	1	11.61	Toms Spe	Luisenstr. 48	Münster	<NULL>	44087	Germany
10250	HANAR	4	7/8/1996	8/5/1996	7/12/1996	2	65.83	Hanari Car	Rua do Paço,	Rio de Ja	RJ	05454-876	Brazil
10251	VICTE	3	7/8/1996	8/5/1996	7/15/1996	1	41.34	Victuailles	2, rue du Cor	Lyon	<NULL>	69004	France
10252	SUPRD	4	7/9/1996	8/6/1996	7/11/1996	2	51.3	Suprêmes	Boulevard Tir	Charleroi	<NULL>	B-6000	Belgium
10253	HANAR	3	7/10/1996	7/24/1996	7/16/1996	2	58.17	Hanari Car	Rua do Paço,	Rio de Ja	RJ	05454-876	Brazil
10254	CHOPS	5	7/11/1996	8/8/1996	7/23/1996	2	22.98	Chop-suey	Hauptstr. 31	Bern	<NULL>	3012	Switzerland
10255	RICSU	9	7/12/1996	8/9/1996	7/15/1996	3	148.33	Richter Su	Starenweg 5	Genève	<NULL>	1204	Switzerland
10256	WELLI	3	7/15/1996	8/12/1996	7/17/1996	2	13.97	Wellington	Rua do Merca	Resende	SP	08737-363	Brazil
10257	HILAA	4	7/16/1996	8/13/1996	7/22/1996	3	81.91	HILARION	Carrera 22 co	San Crist	Táchira	5022	Venezuela
10258	ERNSH	1	7/17/1996	8/14/1996	7/23/1996	1	140.51	Ernst Han	Kirchgasse 6	Graz	<NULL>	8010	Austria
10259	CENTC	4	7/18/1996	8/15/1996	7/25/1996	3	3.25	Centro co	Sierras de Gr	México D	<NULL>	05022	Mexico
10260	OTTIK	4	7/19/1996	8/16/1996	7/29/1996	1	55.09	Ottilies Kä	Mehrheimers	Köln	<NULL>	50739	Germany
10261	QUEDE	4	7/19/1996	8/16/1996	7/30/1996	2	3.05	Que Delíci	Rua da Panifi	Rio de Ja	RJ	02389-673	Brazil
10262	RATTC	8	7/22/1996	8/19/1996	7/25/1996	3	48.29	Rattlesnak	2817 Milton D	Albuquer	NM	87110	USA
10263	ERNSH	9	7/23/1996	8/20/1996	7/31/1996	3	146.06	Ernst Han	Kirchgasse 6	Graz	<NULL>	8010	Austria
10264	FOLKO	6	7/24/1996	8/21/1996	8/23/1996	3	3.67	Folk och fä	Åkergatan 24	Bräcke	<NULL>	S-844 67	Sweden
10265	BLONP	2	7/25/1996	8/22/1996	8/12/1996	1	55.28	Blondel pè	24, place Klé	Strasbou	<NULL>	67000	France
10266	WARTH	3	7/26/1996	9/6/1996	7/31/1996	3	25.73	Wartian H	Torikatu 38	Oulu	<NULL>	90110	Finland
10267	FRANK	4	7/29/1996	8/26/1996	8/6/1996	1	208.58	Frankenve	Berliner Platz	München	<NULL>	80805	Germany
10268	GROSR	8	7/30/1996	8/27/1996	8/2/1996	3	66.29	GROSELLA	5ª Ave. Los P	Caracas	DF	1081	Venezuela
10269	WHITC	5	7/31/1996	8/14/1996	8/9/1996	1	4.56	White Clo	1029 - 12th A	Seattle	WA	98124	USA
10270	WARTH	1	8/1/1996	8/29/1996	8/2/1996	1	136.54	Wartian H	Torikatu 38	Oulu	<NULL>	90110	Finland
10271	SPLIR	6	8/1/1996	8/29/1996	8/30/1996	2	4.54	Split Rail B	P.O. Box 555	Lander	WY	82520	USA
10272	RATTC	6	8/2/1996	8/30/1996	8/6/1996	2	98.03	Rattlesnak	2817 Milton D	Albuquer	NM	87110	USA
10273	QUICK	3	8/5/1996	9/2/1996	8/12/1996	3	76.07	QUICK-Sto	Taucherstraß	Cunewal	<NULL>	01307	Germany
10274	VINET	6	8/6/1996	9/3/1996	8/16/1996	1	6.01	Vins et alc	59 rue de l'Al	Reims	<NULL>	51100	France
10275	MAGAA	1	8/7/1996	9/4/1996	8/9/1996	1	26.93	Magazzini	Via Ludovico	Bergamo	<NULL>	24100	Italy

Products:



Data in Table 'Products' in 'Northwind'

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
17	Alice Mutton	7	6	20 - 1 kg tins	39	0	0	0	1
18	Carnarvon Tig	7	8	16 kg pkg.	62.5	42	0	0	0
19	Teatime Choc	8	3	10 boxes x 12 pie	9.2	25	0	5	0
20	Sir Rodney's M	8	3	30 gift boxes	81	40	0	0	0
21	Sir Rodney's S	8	3	24 pkgs. x 4 piec	10	3	40	5	0
22	Gustaf's Knäcl	9	5	24 - 500 g pkgs.	21	104	0	25	0
23	Tunnbröd	9	5	12 - 250 g pkgs.	9	61	0	25	0
24	Guaraná Fant	10	1	12 - 355 ml cans	4.5	20	0	0	1
25	NuNuCa Nuß-I	11	3	20 - 450 g glasse	14	76	0	30	0
26	Gumbär Gumm	11	3	100 - 250 g bags	31.23	15	0	0	0
27	Schoggi Schok	11	3	100 - 100 g piece	43.9	49	0	30	0
28	Rössle Sauerk	12	7	25 - 825 g cans	45.6	26	0	0	1
29	Thüringer Ros	12	6	50 bags x 30 sau	123.79	0	0	0	1
30	Nord-Ost Mat	13	8	10 - 200 g glasse	25.89	10	0	15	0
31	Gorgonzola Te	14	4	12 - 100 g pkgs	12.5	0	70	20	0
32	Mascarpone F	14	4	24 - 200 g pkgs.	32	9	40	25	0
33	Geitost	15	4	500 g	2.5	112	0	20	0
34	Sasquatch Ale	16	1	24 - 12 oz bottles	14	111	0	15	0
35	Steeleye Stou	16	1	24 - 12 oz bottles	18	20	0	15	0
36	Inlagd Sill	17	8	24 - 250 g jars	19	112	0	20	0
37	Gravad lax	17	8	12 - 500 g pkgs.	26	11	50	25	0
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.5	17	0	15	0
39	Chartreuse ve	18	1	750 cc per bottle	18	69	0	5	0
40	Boston Crab M	19	8	24 - 4 oz tins	18.4	123	0	30	0
41	Jack's New En	19	8	12 - 12 oz cans	9.65	85	0	10	0
42	Singaporean H	20	5	32 - 1 kg pkgs.	14	26	0	0	1
43	Ipoh Coffee	20	1	16 - 500 g tins	46	17	10	25	0
44	Gula Malacca	20	2	20 - 2 kg bags	19.45	27	0	15	0
45	Rogede sild	21	8	1k pkg.	9.5	5	70	15	0
46	Spegesild	21	8	4 - 450 g glasses	12	95	0	0	0
47	Zaanse koeke	22	3	10 - 4 oz boxes	9.5	36	0	0	0
48	Chocolade	22	3	10 pkgs.	12.75	15	70	25	0
49	Maxilaku	23	3	24 - 50 g pkgs	20	10	60	15	0

Region:

Data in Table 'Region' in 'Northwind' ...

RegionID	RegionDescription
1	Eastern
2	Western
3	Northern
4	Southern

Shippers:

Data in Table 'Shippers' in 'Northwind' ...

ShipperID	CompanyName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

Suppliers:

Data in Table 'Suppliers' in 'Northwind'

SupplierID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax	HomePage
1	Exotic Liquids	Charlotte Coo	Purchasing Ma	49 Gilbert	Londo	<NULL>	EC1 4SD	UK	(171) 5	<NULL>	<NULL>
2	New Orleans Ca	Shelley Burke	Order Adminis	P.O. Box	New C	LA	70117	USA	(100) 5	<NULL>	#CAJUN.HTI
3	Grandma Kelly's	Regina Murphy	Sales Represe	707 Oxfo	Ann A	MI	48104	USA	(313) 5	(313) 555	<NULL>
4	Tokyo Traders	Yoshi Nagase	Marketing Mar	9-8 Sekim	Tokyo	<NULL>	100	Japan	(03) 35	<NULL>	<NULL>
5	Cooperativa de	Antonio del Va	Export Admini	Calle del I	Oviedo	Asturias	33007	Spain	(98) 59	<NULL>	<NULL>
6	Mayumi's	Mayumi Ohno	Marketing Rep	92 Setsuk	Osaka	<NULL>	545	Japan	(06) 43	<NULL>	Mayumi's (or
7	Pavlova, Ltd.	Ian Devling	Marketing Mar	74 Rose S	Melbou	Victoria	3058	Australia	(03) 44	(03) 444-6	<NULL>
8	Specialty Biscuit	Peter Wilson	Sales Represe	29 King's	Manch	<NULL>	M14 GSD	UK	(161) 5	<NULL>	<NULL>
9	PB Knäckebröd /	Lars Peterson	Sales Agent	Kaloadag.	Göteb	<NULL>	S-345 67	Sweden	031-98	031-987 6	<NULL>
10	Refrescos Amer	Carlos Diaz	Marketing Mar	Av. das A	Sao Pa	<NULL>	5442	Brazil	(11) 55	<NULL>	<NULL>
11	Heli Süßwaren G	Petra Winkler	Sales Manage	Tiergarte	Berlin	<NULL>	10785	Germany	(010) 9	<NULL>	<NULL>
12	Plutzer Lebensm	Martin Bein	International I	Bogenalle	Frankf	<NULL>	60439	Germany	(069) 9	<NULL>	Plutzer (on t
13	Nord-Ost-Fisch I	Sven Petersen	Coordinator F	Frahmred	Cuxha	<NULL>	27478	Germany	(04721	(04721) 8	<NULL>
14	Formaggi Fortini	Elio Rossi	Sales Represe	Viale Dant	Raver	<NULL>	48100	Italy	(0544)	(0544) 60	#FORMAGG
15	Norske Meierier	Beate Vileid	Marketing Mar	Hatlevegr	Sandv	<NULL>	1320	Norway	(0)2-95	<NULL>	<NULL>
16	Bigfoot Brewerie	Cheryl Saylor	Regional Acco	3400 - 8th	Bend	OR	97101	USA	(503) 5	<NULL>	<NULL>
17	Svensk Sjöföda	Michael Björn	Sales Represe	Brovallav.	Stockh	<NULL>	S-123 45	Sweden	08-123	<NULL>	<NULL>
18	Aux joyeux eccl	Guylène Nodie	Sales Manage	203, Rue	Paris	<NULL>	75004	France	(1) 03.8	(1) 03.83.	<NULL>
19	New England Se	Robb Merchan	Wholesale Acc	Order Prc	Bostor	MA	02134	USA	(617) 5	(617) 555	<NULL>
20	Leka Trading	Chandra Leka	Owner	471 Serar	Singap	<NULL>	0512	Singapor	555-87	<NULL>	<NULL>
21	Lyngbysild	Niels Petersen	Sales Manage	Lyngbysil	Lyngb	<NULL>	2800	Denmark	43844 1	43844115	<NULL>
22	Zaanse Snoepfa	Dirk Luchte	Accounting Ma	Verkoop F	Zaand	<NULL>	9999 ZZ	Netherla	(12345	(12345) 1	<NULL>
23	Karkki Oy	Anne Heikkone	Product Mana	Valtakatu	Lappe	<NULL>	53120	Finland	(953) 1	<NULL>	<NULL>
24	G'day, Mate	Wendy Mackei	Sales Represe	170 Princ	Sydne	NSW	2042	Australia	(02) 55	(02) 555-4	G'day Mate (
25	Ma Maison	Jean-Guy Lau	Marketing Mar	2960 Rue	Montr	Québec	H1J 1C3	Canada	(514) 5	<NULL>	<NULL>
26	Pasta Buttini s.r	Giovanni Giudi	Order Adminis	Via dei Ge	Salern	<NULL>	84100	Italy	(089) 6	(089) 654	<NULL>
27	Escargots Nouv	Marie Delamar	Sales Manage	22, rue H	Montc	<NULL>	71300	France	85.57.(	<NULL>	<NULL>
28	Gai pâturage	Eliane Noz	Sales Represe	Bat. B 3,	Annec	<NULL>	74000	France	38.76.9	38.76.98.	<NULL>
29	Forêts d'érables	Chantal Goulet	Accounting Ma	148 rue C	Ste-Hy	Québec	J2S 7S8	Canada	(514) 5	(514) 555	<NULL>

Territories:

Data in Table 'Territories' in 'Northwind' ...

TerritoryID	TerritoryDescriptior	RegionID
01581	Westboro	1
01730	Bedford	1
01833	Georgetow	1
02116	Boston	1
02139	Cambridge	1
02184	Braintree	1
02903	Providence	1
03049	Hollis	3
03801	Portsmouth	3
06897	Wilton	1
07960	Morristown	1
08837	Edison	1
10019	New York	1
10038	New York	1
11747	Mellvile	1
14450	Fairport	1
19428	Philadelphia	3
19713	Neward	1
20852	Rockville	1
27403	Greensboro	1
27511	Cary	1
29202	Columbia	4
30346	Atlanta	4
31406	Savannah	4
32859	Orlando	4

CustomerCustomerDemo:

Data in Table 'CustomerCustomerDemo' in 'Northwind' ...

CustomerID	CustomerTypeID

CustomerDemographics:

Data in Table 'CustomerDemographics' in 'Northwind' ...

CustomerTypeID	CustomerDesc

LIST OF REFERENCES

1. Zloof, M.M. "Query-by-Example: A Database Language." *IBM System Journal*, vol. 16, (1977): 324-343.

2. Clark, Gard J., Wu, Thomas C. "DFQL: Dataflow Query Language for Relational Databases." *Information & Management* 27 (1994): 1-15.

3. Obasanjo, Dare. "A Comparison of Microsoft's C# Programming Language to Sun Microsystems' Java Programming Language." Http://www.soften.ktu.lt/~mockus/gmcsharp/csharp/c-sharp-vs-java.html. Accessed 10 January, 2005.

4. Bennett, Simon, Skelton, John, and K. Lunn. *Schaum's Outline of UML.* New York: McGraw-Hill, 2001.

5. Wigley, Andy, and M. Sutton. *Microsoft .NET Compact Framework.* Redmond: Microsoft Press, 2003.

6. Yao, Paul, and D. Durant. *.NET Compact Framework Programming with C#.* Boston: Addison-Wesley, 2004.

7. Kontio, Mikko. "Java on Pocket PC Devices." Http://www.informit.com/articles/article.asp?p=344816&rl=1. Accessed 10 January, 2005.

8. DeMichillie, Greg. "SQL CE and Data Access." Http://www.directionsonmicrosoft.com/sample/DOMIS/update/2003/02Feb/0203ncfe_illo1.htm. Accessed 10 January, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Prof Peter J. Denning
   Chairman, Computer Science Department
   Department of Computer Science
   Code CS
   Naval Postgraduate School
   Monterey, California

4. Prof. Thomas W. Otani
   Department of Computer Science
   Code CS/Wq
   Naval Postgraduate School
   Monterey, California

5. Arijit Das
   Department of Computer Science
   Code CS
   Naval Postgraduate School
   Monterey, California

6. Mark A. Evangelista
   TRAC Monterey
   Naval Postgraduate School
   PO Box 8695
   Monterey, California